Juan Nieto
Jose Guivant
Eduardo Nebot
ARC Centre of Excellence for Autonomous Systems (CAS)
The University of Sydney, NSW, Australia
j.nieto@acfr.usyd.edu.au

DenseSLAM: Simultaneous Localization and Dense Mapping

Abstract

This paper addresses the problem of environment representation for Simultaneous Localization and Mapping (SLAM) algorithms.

One of the main problems of SLAM is how to interpret and synthesize the external sensory information into a representation of the environment that can be used by the mobile robot to operate autonomously. Traditionally, SLAM algorithms have relied on sparse environment representations. However, for autonomous navigation, a more detailed representation of the environment is necessary, and the classic feature-based representation fails to provide a robot with sufficient information. While a dense representation is desirable, it has not been possible for SLAM paradigms.

This paper presents DenseSLAM, an algorithm to obtain and maintain detailed environment representations. The algorithm represents different sensory information in dense multi-layered maps*. Each layer can represent different properties of the environment, such as occupancy, traversability, elevation or each layer can describe the same environment property using different representations. Implementations of the algorithm with two different representations for the dense maps are shown.*

A rich representation has several potential advantages to assist the navigation process, for example to facilitate data association using multi-dimensional maps. This paper presents two particular applications to improve the localization process; the extraction of complex landmarks from the dense maps and the detection of areas with dynamic objects. The paper also presents an analysis of consistency of the maps obtained with DenseSLAM. The position error in the dense maps is analyzed and a method to select the landmarks in order to minimize these errors is explained.

The algorithm was tested with outdoor experimental data taken with a ground vehicle. The experimental results show that the algorithm can obtain dense environment representations and that the detailed representation can be used to improve the vehicle localization process.

The International Journal of Robotics Research
Vol. 25, No. 8, August 2006, pp. xxx-xxx
DOI: 10.1177/0278364906067379
©2006 SAGE Publications
Figures appear in color online: http://ijr.sagepub.com

KEY WORDS—simultaneous localization and mapping (SLAM), EKF-SLAM, autonomous mapping

1. Introduction

Map building is the process of obtaining a representation of the environment using external sensory information. A mobile robot can use a wide variety of sensors to obtain information about the environment. Typical sensors are cameras, lasers, radars and sonars (Montemerlo and Thrun 2004; Jose and Adams 2005). Since the observations are taken from sensors on-board the robot, an estimate of the robot pose is necessary in order to build the map.

Building a representation of the environment with an autonomous robot, given the robot pose (Elfes 1989b; Kuipers and Byun 1991; Devy et al. 1995; Jensfelt and Kristensen 2001) is considered a problem solved today. However, the robot pose will in general be unknown, and the mapping process will have to be done simultaneously with the robot localization.

Simultaneous localization and mapping algorithms provide a means to localize the robot using observations of the environment. Since SLAM algorithms build a map simultaneously with the localization process, they do not require an *a priori* map. The SLAM problem was introduced eighteen years ago (Smith, Self, and Cheeseman 1987). This seminal paper explained the need for concurrently estimating the vehicle pose and map position, and presented a stochastic solution based on the use of an Extended Kalman Filter (EKF). These foundations have served as the basis for most of the current solutions. The vehicle takes observations of the environment using sensors such as range lasers, and identifies features that are incorporated in the map as landmarks. EKF-SLAM maintains a state vector with the robot pose and landmark positions. The map is then constructed as an on-line data fusion problem and an estimate of uncertainties in the robot pose and landmark locations is maintained.

One of the issues in SLAM algorithms is the computational cost, which grows quadratically with the number of landmarks in the map, $O(n^2)$. To overcome this problem, efficient solutions have appeared (Knight, Davison, and Reid 2001; Williams 2001; Guivant 2002). By creating local maps, only the landmarks in a local region have to be updated. The computational cost is then quadratic with the number of landmarks in the vehicle's vicinity. The local maps are periodically fused with the global map. Sub-optimal SLAM algorithms that are able to reduce the computational cost to constant time, $O(1)$ have also appeared (Bailey 2002; Bosse et al. 2003).

An important issue with stochastic SLAM is environment modeling. SLAM maps are typically sparse and are built up of isolated landmarks observed in the environment. This makes the representation appropriate only for localization purposes.

The next generation of autonomous systems will be required to operate in more complex environments. A sparse representation formed only by isolated landmarks will in general not fulfill the necessities of an autonomous vehicle, and a more detailed representation will be needed for tasks such as place recognition or path planning. Furthermore, not only is a dense representation of the environment required, but also an algorithm that is able to obtain *multi-layered maps* (Lacroix et al. 2002), where each layer represents a different property of the environment, such as occupancy, traversability, elevation, etc.

This paper presents DenseSLAM (Nieto, Guivant, and Nebot 2004b). The algorithm is able to construct detailed multi-layer descriptions of the environment. It combines feature maps with other dense metric sensory representation. DenseSLAM is based on the Hybrid Metric Maps (HYMMs) introduced by Guivant et al. (2004). The global features map is partitioned into a set of connected local regions, which provides a reference for a detailed multi-dimensional description of the environment. The HYMM framework permits the combination of efficient feature-based SLAM algorithms for localization and dense mapping representations such as occupancy grids (OGs).

A more comprehensive environment representation allows the vehicle localization process to be improved. This paper presents two cases where the dense maps are used to aid vehicle localization. The first one shows how to detect and incorporate complex landmarks that cannot be detected from only one vantage point. The second one shows how the dense representation can be used to detect regions with potential dynamic objects. This information will help to prevent the vehicle from using dynamic objects for the localization process.

The format of this paper is as follows. Section 2 introduces the problem of feature-based SLAM. Section 3 explains the solution presented here for the DenseSLAM problem. Section 4 discusses consistency in DenseSLAM. Section 5 shows different possible representations to be used for the dense maps. Section 6 presents applications of dense mapping and Section 7 presents experimental results. Finally, conclusions are presented in Section 8.

2. Stochastic SLAM

To obtain a representation of the world, robots must possess sensors able to acquire external information. In addition, a robot usually has sensors to predict its position. Unfortunately, all sensors are subject to errors (sensor noise). Thus any solution to the SLAM problem will need to incorporate the sensor's noise into the models in order to be able to fuse data taken at different times.

Probabilistic methods are the common factor among SLAM algorithms available today (Thrun 2002). This is because they are able to incorporate the sensors noise into the models. Furthermore, because the observations of the external world are taken relative to the vehicle position, the uncertainty accumulated over time in the robot pose will be propagated to the map. Therefore, the errors in the vehicle position will be correlated with the map. Probabilistic methods are also able to model the correlations between vehicle pose and the map.

Finding a solution to the stochastic SLAM problem involves evaluating the next probability distribution.

$$P(\mathbf{x}_k | \mathbf{Z}^k, \mathbf{U}^k, \mathbf{x}_0) \tag{1}$$

where $\mathbf{x}_k$ depicts the variables used to represent the vehicle pose $\mathbf{x}_k^v$ and the map position $\mathbf{x}_k^m$ at time k.

$$\mathbf{x}_k = \begin{bmatrix} \mathbf{x}_k^v \\ \mathbf{x}_k^m \end{bmatrix} \tag{2}$$

$\mathbf{Z}^k$ represents the set of observations received until time k.

$$\mathbf{Z}^k = [\mathbf{z_0}, \mathbf{z_1}, ..., \mathbf{z_k}]^T \tag{3}$$

$\mathbf{U}^k$ represents the set of control inputs received until time k.

$$\mathbf{U}^k = [\mathbf{u_0}, \mathbf{u_1}, ..., \mathbf{u_k}]^T \tag{4}$$

and $\mathbf{x}_0$ represents the initial information.

The recursive EKF-SLAM is performed as a sequence of predictions and updates.

The predicted mean and covariance are calculated as:

$$\hat{\mathbf{x}}^- = \begin{bmatrix} \hat{\mathbf{x}}_v^- \\ \hat{\mathbf{x}}_m^- \end{bmatrix} = \begin{bmatrix} \mathbf{f}(\hat{\mathbf{x}}_v, \mathbf{u}) \\ \hat{\mathbf{x}}_m \end{bmatrix} \tag{5}$$

$$\mathbf{P}^- = \nabla\mathbf{f_x}\mathbf{P}\nabla\mathbf{f_x}^T + \nabla\mathbf{f_u}\mathbf{Q}\nabla\mathbf{f_u}^T \tag{6}$$

where $\mathbf{f}$ represents the relation between the past estimated robot pose and the predicted according to the new control inputs received $\mathbf{u}$. Since the map is assumed to be formed by stationary landmarks, the map position does not change during the prediction step. $\mathbf{P}$ represents the covariance matrix and $\mathbf{Q}$ the covariance matrix of the noise in the control inputs. The Jacobians in eq. (6) are:

$$\nabla\mathbf{f_x} = \left.\frac{\partial\mathbf{f}}{\partial\hat{\mathbf{x}}}\right|_{(\hat{\mathbf{x}},\mathbf{u})} \tag{7}$$

$$\nabla\mathbf{f_u} = \left.\frac{\partial\mathbf{f}}{\partial\mathbf{u}}\right|_{(\hat{\mathbf{x}},\mathbf{u})}$$

On arrival of a new observation, the state vector is updated as:

$$\hat{\mathbf{x}}^{+} = \hat{\mathbf{x}}^{-} + \mathbf{W}\nu_i \quad (8)$$

$$\mathbf{P}^{+} = \mathbf{P}^{-} - \mathbf{W}\mathbf{S}\mathbf{W}^{T} \quad (9)$$

where $\mathbf{W}$ is the Kalman gain, ν_i is the innovations vector and $\mathbf{S}$ is the innovation covariance matrix.

3. DenseSLAM

Mapping techniques that are able to handle vehicle uncertainty such as EKF-SLAM are not able to obtain dense representations due to the extremely high computational burden involved. On the other hand, mapping algorithms that are able to obtain detailed representations such as Occupancy Grids (Elfes 1989a), are known to have problems coping with vehicle pose uncertainty. The concept of DenseSLAM was introduced in Nieto, Guivant, and Nebot (2004b) as

*The process of simultaneous vehicle localization and **dense** map building.*

DenseSLAM is then a more ambitious problem than classic feature-based SLAM. A solution for DenseSLAM will have to deal with computational and consistency issues, arising from the dual purpose of trying to obtain a dense representation while simultaneously doing localization.

This section explains the framework on which DenseSLAM is built, the *Hybrid Metric Maps*.

3.1. The HYbrid Metric Maps (HYMMs)

The Hybrid Metric Maps (HYMMs) algorithm (Guivant et al. 2004; Nieto, Guivant, and Nebot 2004b) presents a novel solution for addressing the mapping problem with unknown robot pose. The HYMM is a mapping algorithm that combines feature maps with other metric sensory information. The approach permits the localization of the robot and at the same time constructs a detailed environment representation (DenseSLAM).

Rather than incorporating all the sensory information into a global map, the algorithm maintains a features map and represents the rest of the sensed data in local maps defined relative to the feature positions. A joint state vector with the vehicle pose and the feature positions is maintained and the dense maps are stored in a separate data structure. When new observations are received, the state vector is augmented with the feature positions and the rest of the information is fused into the local regions. The main difference between feature-based SLAM and DenseSLAM is that feature-based SLAM incorporates the features into the map and neglects the rest of the information, whereas DenseSLAM has the ability to maintain all the sensory information to build a detailed environment representation.

The algorithm works as follows. When the robot starts to navigate, it will extract features from the environment that will be incorporated in the state vector. The feature map will be used to partition the global map into smaller regions, Figure 1 illustrates this process. The dense sensory information will be represented in these local regions. Figure 2 shows a hypothetical dense map. The figure shows the division of the global map into smaller regions and the dense multi-layer maps obtained by DenseSLAM. Each of these layers depicts different environment properties. The global dense map consists of a set of local maps defined relative to the feature positions. Figure 3 shows a basic diagram of the algorithm. (See also Extension 1 for a movie that illustrates how the algorithm works.)

The main characteristic of DenseSLAM is the local representation used to fuse the dense information. The motivation behind the relative representation is to reduce correlations between states. Using this relative representation, the states represented in a local frame become strongly correlated and the states represented in different frames become weakly correlated. This is the key which allows the decorrelation of the dense maps with the rest of the system making the representation tractable.

It will be shown that the estimation of the features map and the localization process obtained with DenseSLAM are optimal in respect of estimates using only feature-based SLAM. Moreover, the estimates can be improved if the dense information is used to assist the SLAM process (e.g., during the data association process).

The next sections explain the fundamental aspects of the algorithm.

3.2. Fundamental Principle of DenseSLAM

Since the observations of the world are taken relative to the vehicle pose, any environment representation created will be correlated with the vehicle pose. Augmenting the state vector with all the information rendered by the sensors and maintaining the correlations is infeasible because of the computational burden involved. Therefore, DenseSLAM incorporates a set of landmarks in the state vector and the rest of the sensed data is decorrelated and stored in a separate data structure.

The approximation made by the algorithm consists of representing the dense information in the local regions without including the correlations between the locally represented information and the rest of the system. These correlations will be zero only when there is full correlation between the local property (expressed in global coordinates) and the features that define the respective local frame (assuming the same uncertainty magnitude), so their relative positions are perfectly known. Although it can be proved that in a SLAM process the map becomes fully correlated in the limit (Gibbens, Dissanayake, and Durrant-Whyte 2000), in practice only high correlation is achieved. However, it can be demonstrated that the assumptions made by the HYMM framework are, in prac-

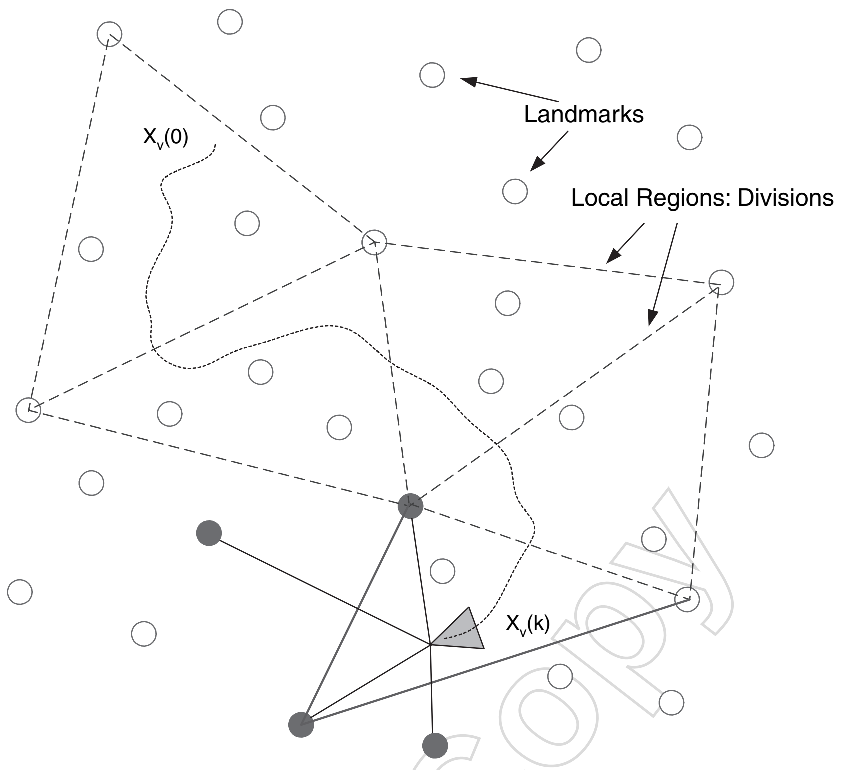


Fig. 1. Landmarks map (“o”) and a particular partition of the global map in local regions. As shown, not all the landmarks are needed as vertex points in the regions definition.

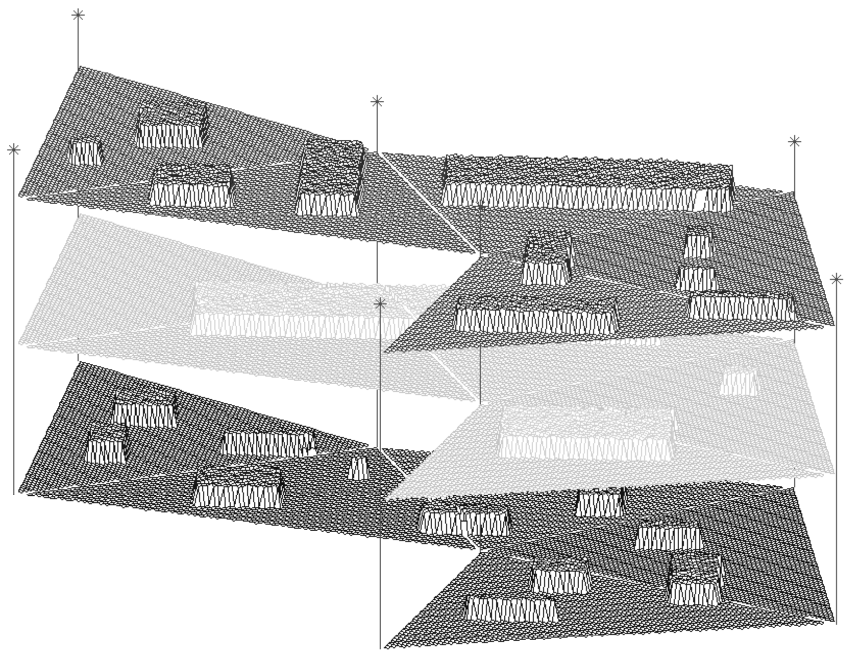


Fig. 2. Hypothetical multi-layer dense map. The “*” represent the landmark positions and the map layers depict different environment properties captured by the sensors.

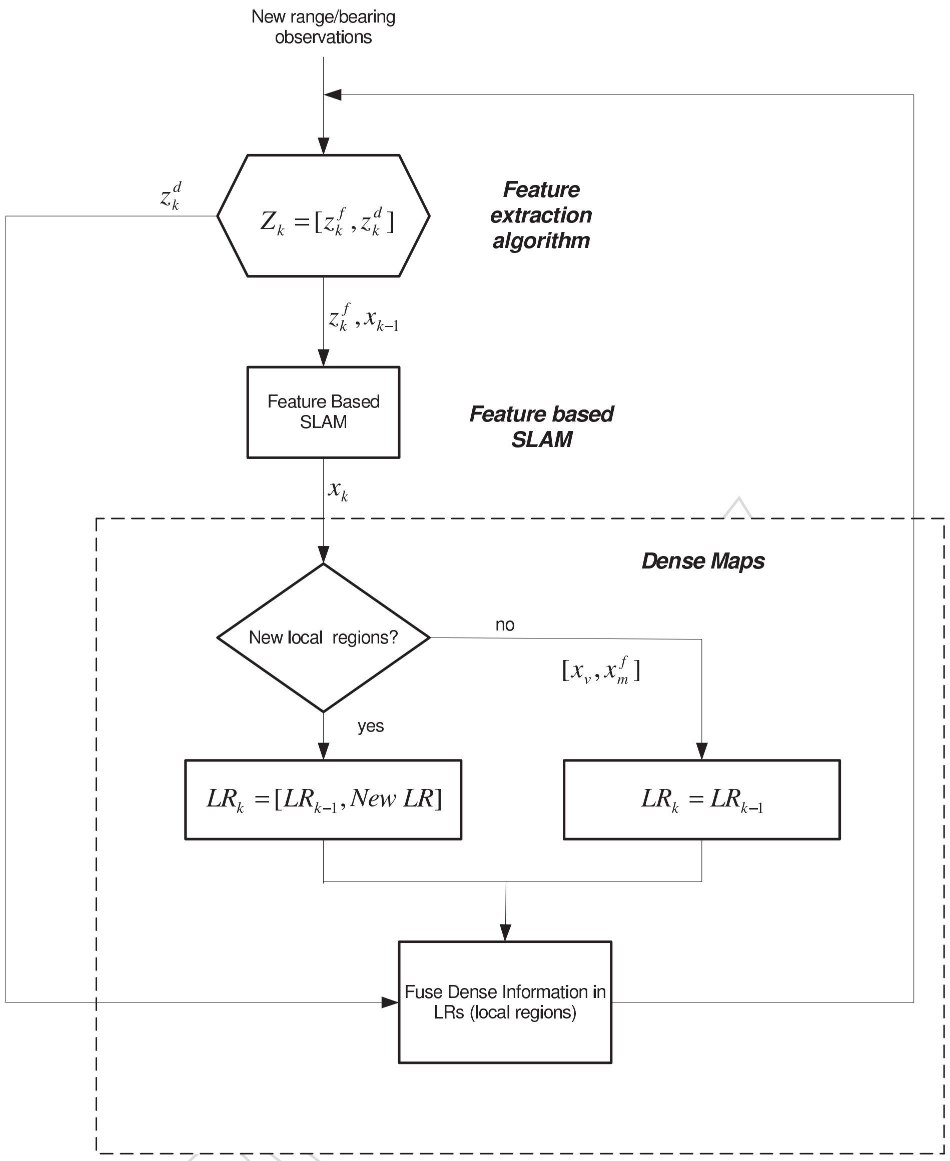


Fig. 3. HYMM algorithm flow diagram. When a sensor frame is obtained, first a feature extraction algorithm is applied and the features extracted are added to the feature-based SLAM. Then the algorithm looks for new local regions (LR) and fuses all the sensory information in the respective local frames. $\mathbf{z}_k^f$ represents the observations associated with features and $\mathbf{z}_k^d$ the rest of the observations (dense maps). $\mathbf{x}_v$ represents the vehicle position and $\mathbf{x}_m^f$ the feature map.

tice, very good approximations for SLAM problems. The next paragraphs explain two well known properties of SLAM that justify the approximations made in the HYMMs.

1. *Geographically close objects have high correlation:* If a set of observed objects are geographically close from the vehicle viewpoint, then the error resulting from the vehicle pose uncertainty will be a common component of these estimated objects' positions. This is a typical situation in SLAM where the vehicle accumulates uncertainty in its estimated position and incorporates observations that are used to synthesize a map. Because of this the estimates of objects that are geographically close will present similar uncertainties (high cross-correlations). Any update of a particular object will imply a similar update of any object sufficiently close to the first one. Figure 4 shows an example of a typical SLAM map. The figure shows a landmarks map with its uncertainty bounds. It can be seen that the uncertainty is very similar in landmarks that are geographically close.

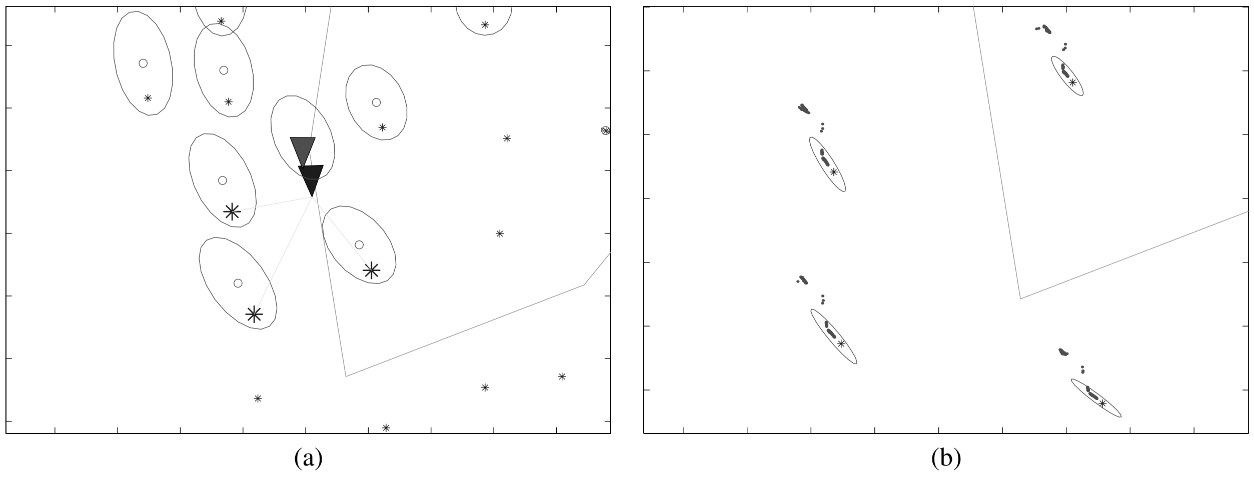


Fig. 4. Map Correlation: The figures show that geographically close objects possess similar uncertainty. Figure (a) shows how the landmarks that are being observed have similar uncertainty to the robot pose. (b) Shows how the estimated landmarks' means are updated after the vehicle closes the loop. The dots represent landmark position estimates over time. High correlation in geographically close objects is one of the SLAM characteristics; because the vehicle will observe close objects at similar instants it will propagate similar uncertainty to the objects.

2. *The relative representation stores close objects in local coordinate frames and then permits the reduction of correlation to the rest of the map (Guivant and Nebot 2003):* Assume a landmark can be represented in a local frame in the following way:

$$\mathbf{x}^L = \mathbf{h}(\mathbf{x}_v, \mathbf{z}, \mathbf{x}_b) \tag{10}$$

where $\mathbf{x}^L$ represents the relative landmark position, $\mathbf{x}_v$ the vehicle position, $\mathbf{z}$ the observations and $\mathbf{x}_b$ the position of the landmarks that define the local frame (base landmarks).

Taking into account that the observation errors are independent of the vehicle position (as it is always assumed in SLAM), the cross-correlation between the vehicle and the landmark in the local frame will be:

$$\mathbf{P}_{vL} = \mathbf{P}_{vv}\nabla\mathbf{h}_{\mathbf{x}_v}^T + \mathbf{P}_{vb}\nabla\mathbf{h}_{\mathbf{x}_b}^T \tag{11}$$

where $\mathbf{P}_{vv}$ represents the vehicle states covariance, $\mathbf{P}_{vb}$ the cross-correlation between the vehicle states and the base landmarks position estimated and $\nabla\mathbf{h}_{\mathbf{x}_i} = \frac{\partial\mathbf{h}}{\partial\mathbf{x}_i}$ is the Jacobian matrix of $\mathbf{h}$ with respect to the state $\mathbf{x}_i$.

Taking for example the one dimensional case, eq. (10) becomes:

$$x^L = h(x_v, z, x_b) = x_v + z - x_b \tag{12}$$

Applying eq. (12) to (11):

$$\mathbf{P}_{vL} = \mathbf{P}_{vv}(1) + \mathbf{P}_{vb}(-1) = \mathbf{P}_{vv} - \mathbf{P}_{vb} \tag{13}$$

Equation (13) shows that if the magnitudes of $\mathbf{P}_{vv}$ and the covariance of the base landmarks $\mathbf{P}_{bb}$ are similar, when the robot is highly correlated with the base landmarks there will be almost no correlation between the robot position and the local landmark ($\mathbf{P}_{vL}$) and then no correlation between the relative landmark and the rest of the map. Since the relative and the base landmarks are geographically close, whenever the robot observes the local landmark it will be highly correlated with the base landmarks. This fact will reduce the correlation between the local landmark and the robot and therefore the correlation between the local landmark and the rest of the map will be reduced as well.

A more direct way of observing the decorrelation effect is by evaluating the cross-correlation between a landmark in the global frame with a landmark represented in a local frame and comparing this with the cross-correlation of the same two landmarks, both represented in the global frame. In a similar manner to eq. (11), the cross-covariance matrix between the j-th landmark and a locally represented landmark can be evaluated in the following way:

$$\mathbf{P}_{jL} = \mathbf{P}_{jb}\nabla\mathbf{h}_{\mathbf{x}_b}^T + \mathbf{P}_{jG}\nabla\mathbf{h}_{\mathbf{x}^G}^T \tag{14}$$

where the prefix j means the j-th landmark, b the base landmarks that define the local frame, L the locally represented landmark and G the position of the local landmark in the global frame. Then given $\mathbf{P}_{jb}$, $\mathbf{P}_{jG}$ and the transforming function from the global to the local frame $\mathbf{h}$, it is possible to evaluate the cross-correlation between the local landmark and the j landmark in the map. Although the effect of decorrelation happens regardless of the particular local representation used, finding an expression to demonstrate that $\mathbf{P}_{jL} << \mathbf{P}_{jG}$ will be dependent on the local representation used. Equation (14) shows that for a particular local representation $\mathbf{h}$, the decorrelation effect between the local object and the rest of the map will depend on the cross correlation between the rest of the map and the base landmarks and the cross correlation between the rest of the map and the local represented object in the global frame.

A numerical example is shown next to illustrate the decorrelation effect using local representation. In the example presented, three landmarks are used to define the axes of a local region.

EXAMPLE 1. Figure 5(a) shows the simulation environment utilized to illustrate the decorrelation effect. In order to show the decorrelation effect, a local region is defined using three landmarks and another landmark is locally represented in this local frame. The i-th landmark is then represented in the local frame after a few observations which ensures high correlation with the base landmarks. After that, the landmark is not observed again, as it actually occurs with the dense sensory information (it is assumed that is not possible to observe exactly the same point of the environment more than once). The blue solid line in Figure 5(a) shows the local frame axis and the red dotted line the local landmark position $\mathbf{x}_{Li}$. The cyan dashed line joins the local represented landmark $\mathbf{x}_{Li}^{L}$ with a global landmark $\mathbf{x}_{Lj}$. The cross-correlation between $\mathbf{x}_{Lj}$ and the local i-th landmark will be evaluated when the i landmark is represented in the local frame $\mathbf{x}_{Li}^{L}$ and when it is represented in the global frame $\mathbf{x}_{Li}^{G}$.

Figure 5(b) shows the evolution in time of the correlation coefficient of the i landmark represented in global coordinates $\mathbf{x}_{Li}^{G}$, with the landmarks used to define the local frame. The solid line depicts the cross-correlation in the east axis and the dashed line in the north axis. The different colors represent the cross-correlation with the different base landmarks. As can be seen, the landmark $\mathbf{x}_{Li}^{G}$ possesses high correlation with the base landmarks. This is due to the geographical proximity between the landmarks. Figure 5(c) shows the correlations between $\mathbf{x}_{Li}^{G}$ and $\mathbf{x}_{Lj}$ in red, and the correlations between the j landmark and the landmark i represented in the local frame $\mathbf{x}_{Li}^{L}$ (eq. (14)) in blue. The correlation was reduced from almost one when the landmark was represented in global coordinates, to almost zero when the landmark was represented in the local frame.

Finally Figure 5(d) shows the variance of the landmark i. The blue line depicts the variance when the landmark is in the local frame and the red line when it is in global. Because of the high correlation between $\mathbf{x}_{Li}^{G}$ and the base landmarks, the uncertainty in their relative position is very low, and so is the variance of $\mathbf{x}_{Li}^{L}$.

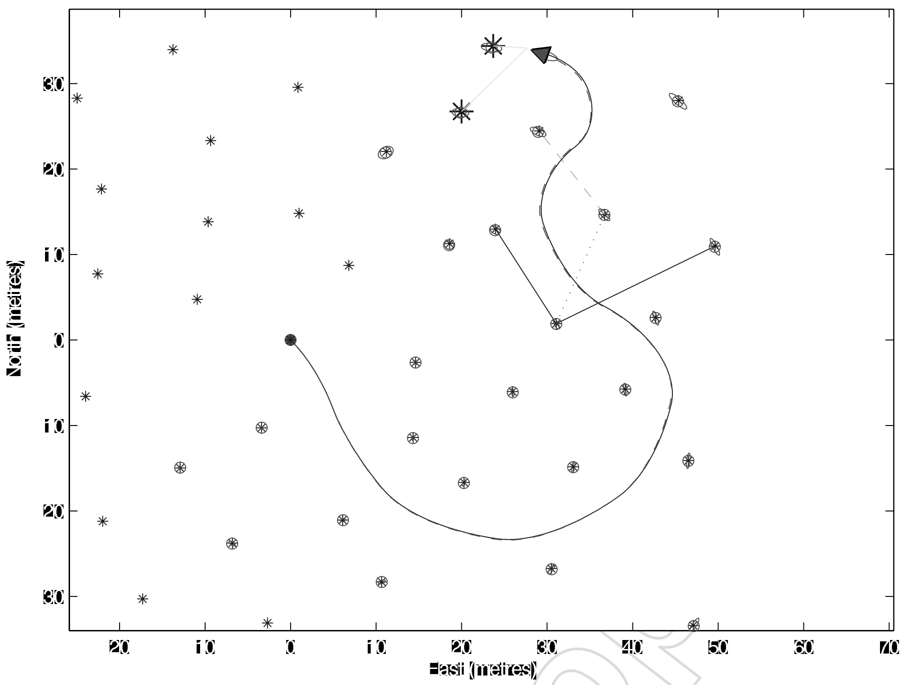
In summary the relative representation used by DenseSLAM permits the local represented information to be decorrelated with the rest of the system. This permits the incorporation of more information without increasing the computational cost.

3.3. Relative Representation

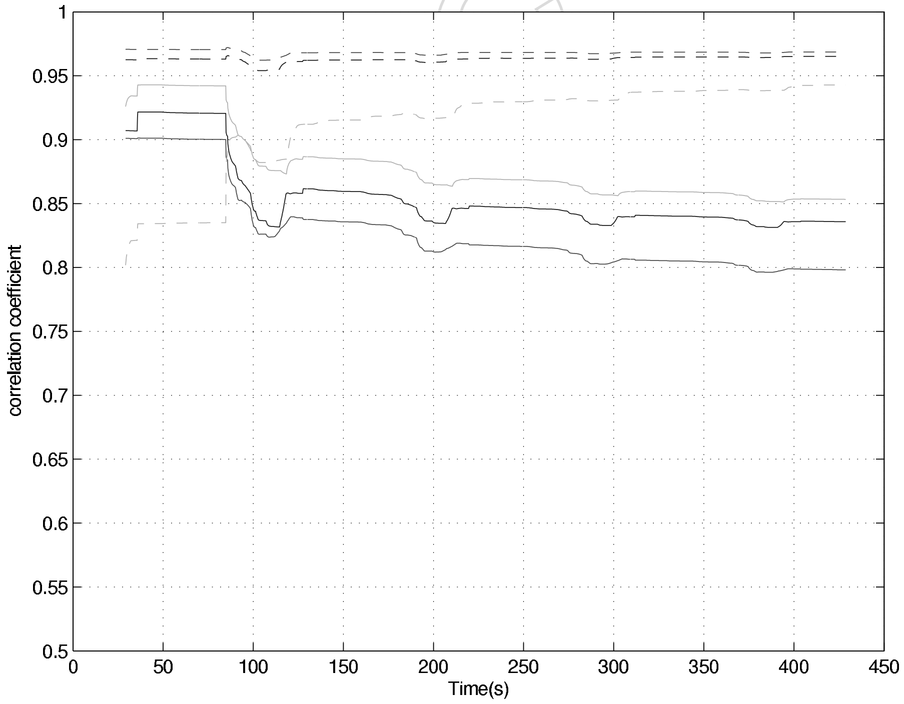
This section discusses different ways of defining the local frames and explains the one used in this paper.

The most common relative representation employs two landmarks to represent the local frames (Guivant and Nebot 2002b) and defines rectangular local regions. The rectangular representation has an important shortcoming when used as the local regions in DenseSLAM; it presents problems in terms of map coverage. Figure 6(a) shows a hypothetical map division using rectangular local regions. As seen in the figure, some local regions overlap. In addition, some areas of the map are not covered by any local region. A solution for covering the whole area would be to overlap all the regions. However this would be inefficient in terms of computation and memory, and also will add difficulty to the global map recovery task. A better solution for covering the whole area in an efficient manner will be to define all the local regions in a form such that they share a common axis, thus avoiding overlapping or gaps between different local maps. Figure 1 shows a possible map division using three landmarks for the bases. Unlike the rectangular representation, the angle between the axes is not restricted to a particular value, the angle is defined by the landmark positions, and can be different for different regions. In this case the global map is partitioned into *local triangular regions* (LTRs). Because the local regions always have common vertices, the whole area can be covered without overlapping regions. Note that this is not a particular property of the triangular division, it is actually a consequence of not confining the angle between the axes to a fixed value, so avoiding the restriction of the local maps to homogeneous regions.

More than three landmarks could be used to define local regions, presenting similar characteristics to the LTRs in terms of map coverage. Figure 6(b) shows an example of a map division using more than three landmarks; four and five landmarks were used to delimit the regions. Using more than three landmarks will imply defining more than one frame per local region, which will give more robustness to the representation but will add complexity to the implementation. The complexity arises from the larger number of landmarks needed to form a region and the repetition of information in different frames. The representation used in this paper employs three landmarks to delimit the local regions. This is the most efficient representation for our purpose, since it involves the smallest number of landmarks while assuring full coverage of the area.

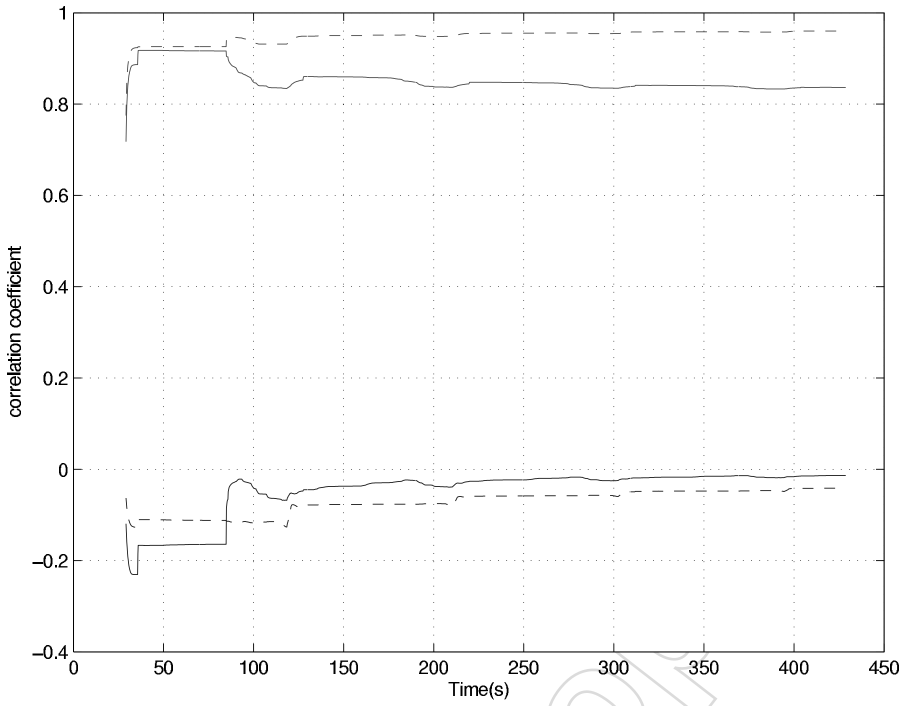


(a)

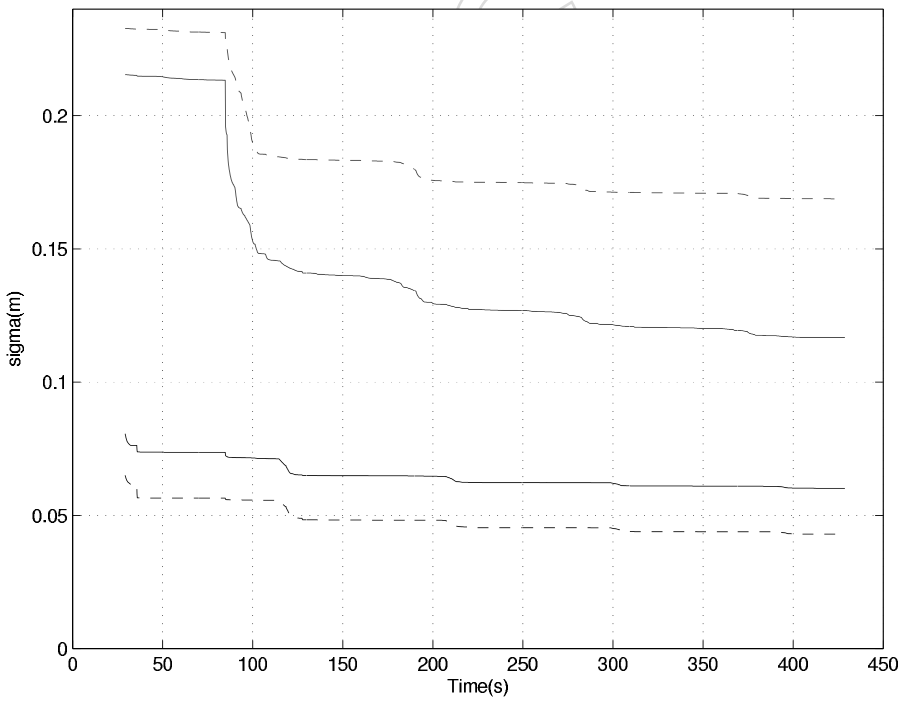


(b)

Fig. 5. Decorrelation effect: (a) shows a zoom of the navigation environment where three landmarks were used to define a local region and one landmark $\mathbf{x}_{Li}$ (red dashed line) was represented in this local frame. (b) Shows the correlation between the landmark i represented in the global frame and the base landmarks. *(Continues on next page)*

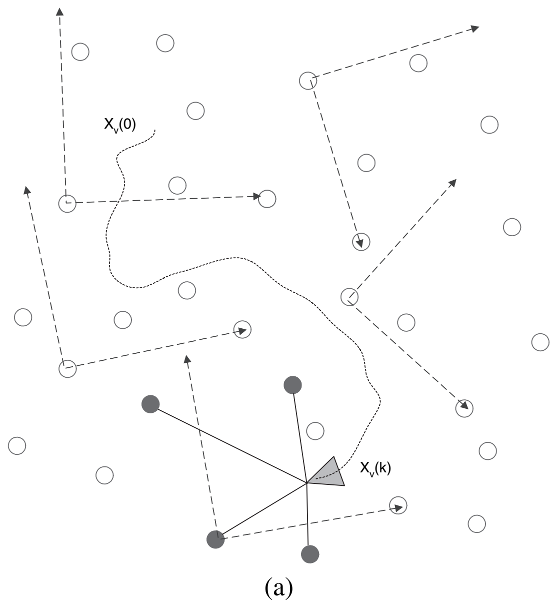


(c)



(d)

Fig. 5. Decorrelation effect: *(Continued from previous page)* (c) Shows the decorrelation effect; when the landmark i is represented in local coordinates (blue line) the cross-correlation with other landmarks is considerably reduced in respect to the correlation between other landmarks and the landmark i in global coordinates (red line). (d) Shows the landmark deviation when it is represented in local (blue line) and global (red line) coordinates.



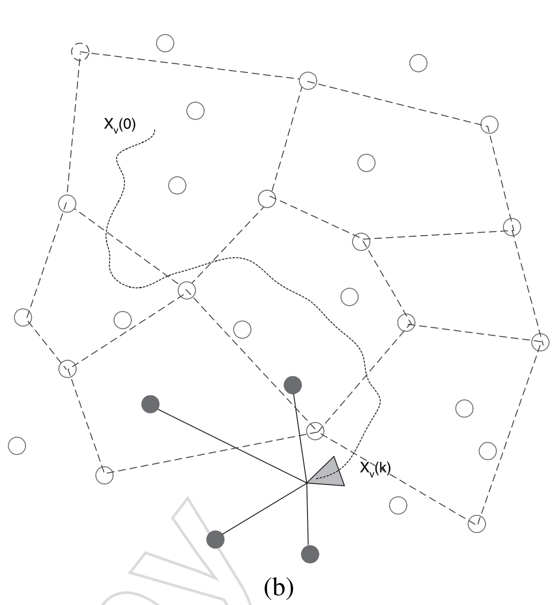


Fig. 6. Figure (a) shows an example of a map division using rectangular regions; (b) shows a map division using four and five landmarks to delimit the regions.

3.3.1. *Local Triangular Regions (LTRs)*

Figure 7 shows an example of a LTR. Any point that belongs to a LTR can be characterized by a convex linear combination of the three vertex points (landmark positions) associated with this subregion. In Figure 7 a LTR Ω_i is defined by the vertex points $\{\mathbf{L}_{i,1}, \mathbf{L}_{i,2}, \mathbf{L}_{i,3}\}$. A local coordinate frame is defined based on the three vertex points according to:

$$\begin{aligned} \mathbf{a}_i &= \mathbf{L}_{i,2} - \mathbf{L}_{i,1} = [a_x, a_y]^T \\ \mathbf{b}_i &= \mathbf{L}_{i,3} - \mathbf{L}_{i,1} = [b_x, b_y]^T \end{aligned} \tag{15}$$

Any point that belongs to Ω_i can be expressed in the global frame as:

$$\begin{aligned} \mathbf{x} &= \mathbf{L}_{i,1} + \alpha \cdot \mathbf{a}_i + \beta \cdot \mathbf{b}_i \\ &= (1 - \alpha - \beta) \cdot \mathbf{L}_1 + \alpha \cdot \mathbf{L}_{i,2} + \beta \cdot \mathbf{L}_{i,3} \end{aligned} \tag{16}$$

$$\begin{aligned} \alpha > 0, \quad \beta > 0 \\ \alpha + \beta \le 1 \\ \forall \ \mathbf{x} \ \backslash \ \mathbf{x} \in \Omega_i \end{aligned}$$

Furthermore any function of the global position $\mathbf{x}$ can also be locally defined as a function of the local coordinates of $\mathbf{x}$ and the vertices points:

$$z = f(\mathbf{x}) = f(\mathbf{L}_{i,1} + \alpha \cdot \mathbf{a}_i + \beta \cdot \mathbf{b}_i) \tag{17}$$

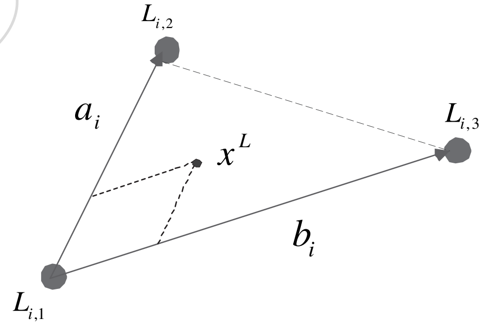


Fig. 7. Detail of an individual LTR defined by three vertex points, $\{L_{i,1}, L_{i,2}, L_{i,3}\}$ and the direction vectors, $\{\overline{a_i}, \overline{b_i}\}$.

Using (15) and (16) an expression for the local coordinates (α, β) can be derived.

$$\begin{aligned} \alpha &= \frac{(x_y - L_{1y})d_{13x} - (x_x - L_{1x})d_{13y}}{d_{13x}d_{12y} - d_{13y}d_{12x}} \\ \beta &= \frac{(x_x - L_{1x})d_{12y} - (x_y - L_{1y})d_{12x}}{d_{13x}d_{12y} - d_{13y}d_{12x}} \end{aligned} \tag{18}$$

where $\mathbf{x} = [x_x, x_y]$ are the global coordinates of the local point and $\mathbf{d}_{ij} = \mathbf{L}_j - \mathbf{L}_i = [\mathbf{d}_{ijx}, \mathbf{d}_{ijy}]^T$. In general, any local point can be expressed as:

$$\mathbf{x}^L = [\alpha, \beta]^T = \mathbf{h}(\mathbf{x}_v, \mathbf{z}, \mathbf{L}) \tag{19}$$

where $\mathbf{x}^L$ represents the local state, $\mathbf{x}_v$ the robot pose, $\mathbf{z}$ the observations and $\mathbf{L}$ the set of landmarks that defines the local region.

Assume a vehicle is exploring the environment and measuring different properties such as soil salinity, humidity, terrain occupancy, etc. Not necessarily all the sensory information has to be used for the robot localization process. Nevertheless, it may be desired to maintain all the information for additional tasks, such as path planning, or for example the detection of a particular property in the environment. The classic SLAM algorithm uses features extracted from sensed data for the localization process and neglects the rest of the information. DenseSLAM also uses a features map, but in addition the algorithm integrates all the sensory information using local representation. The dense maps obtained consist of map layers, where each layer represents the information coming from different sensors, or the same sensor information represented in different forms.

It is worth making clear that the framework is not restricted to any particular representation for the local maps. The only restriction is that the representation used needs to be defined relative to the feature positions.

3.4. Base Landmarks Selection

To minimize the error in the position of the dense information, the framework establishes some conditions that have to be attained by potential landmarks before forming the vertices of a LTR. This section presents these conditions.

- *The landmarks that define a LTR must have high correlation*. As explained in Section , DenseSLAM uses the fact that in SLAM geographically close objects possess high correlation and so their relative position is almost independent of the vehicle pose uncertainty. This makes the locally represented objects have very weak correlation with the rest of the system which allows the algorithm to neglect these correlations. In particular, because the locally represented entities are not used directly for the localization process, the framework does not introduce any inconsistency at all in the system (Section 4.1 will explain this aspect in more detail).

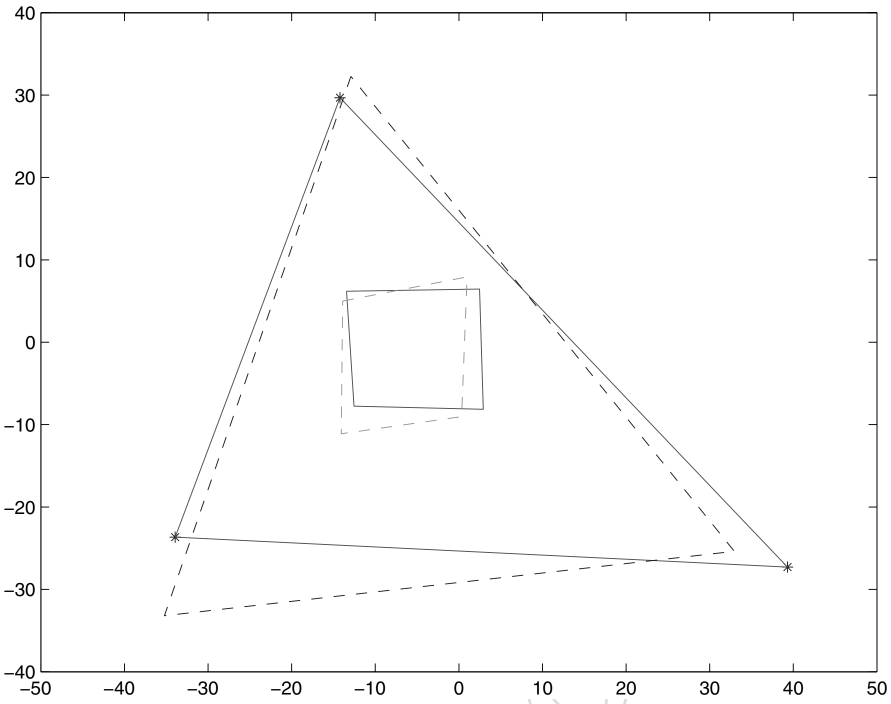
  The ideal situation for the algorithm is when all landmarks and objects in a common local region are fully correlated and present the same uncertainty values. In this case their relative position is perfectly known (zero uncertainty) and the correlation between the locally represented objects and the rest of the system will be zero. The higher the cross-correlation among objects in the global frame, the lower the uncertainty of their relative position.

  Therefore, it can be deduced that the higher the correlation between the base landmarks and the locally represented objects (local map), the better the quality of the local map. In particular, since the axes of the local frame are defined based on the relative position of the base landmarks (Figure 7) it is also important to have high correlation among those landmarks that form a base. High correlation will prevent deformations in the frame after the landmark positions are updated. Figure 8 clarifies this with an example. In Figure 8(a) the region is defined with three landmarks that do not possess high correlation, and an object is defined relative to this base. After the landmarks are updated, the local region is misshapen and so is the object. Figure 8(b) shows the same example, but now the region is formed after the landmarks have achieved high correlation. After the update, the local region has moved without major changes in shape. This is the consequence of having low/high correlation among the base landmarks.

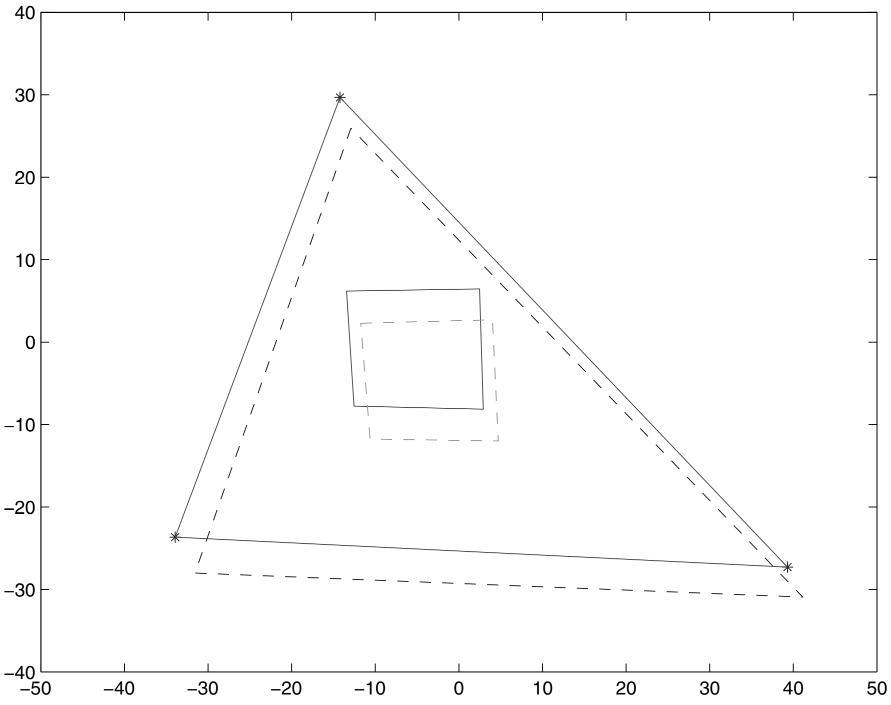
- *The landmarks must also be geometrically well conditioned (to avoid degeneration)*. This condition is related to the geometry of the local region. Before defining a local region, the angle between the local axes has to be checked. Figure 9(a) shows an ill conditioned LTR. This case is referred to as *ill conditioned* because two axes are “too close” to each other, a situation that could make the triangle flip over. Deciding when the axes are too close or when a region is ill conditioned is a process that involves not only the cross correlation between the landmarks but also the individual uncertainty. With the individual uncertainty and the correlations, it can be predicted how much a landmark can change its position after an update, and then predict if any potential region could flip over. An easy and conservative test can be done by considering the landmarks to be independent (no correlation) and then comparing the uncertainty bounds of the landmarks against the opposite axis. For example, in Figure 9(a), if the uncertainty bound (ellipse) of L_2 has contact with its opposite axis (formed by L_1 and L_3), then the region could flip over. In the example presented, the uncertainty bounds of L_2 have contact with the opposite axis, and so the region is said to be *ill conditioned*. This is a very conservative test, since the fact that there is correlation between landmarks will restrict the relative movements.

- *New regions should not overlap with old regions*: The last condition to check is that the regions do not overlap to avoid repetition of the information. Figure 9(b) shows an example of two overlapped regions.

Having verified all these conditions, a new local region can be created and a new local dense map will be built using an adequate representation.

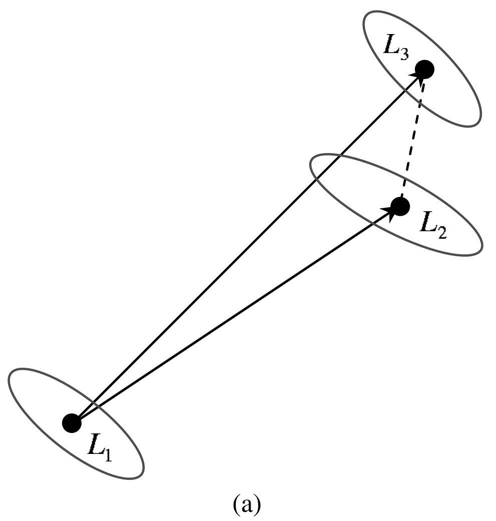


(a)



(b)

Fig. 8. Figure (a) shows a LTR and an object represented locally for the case where the landmarks possess low correlation. The solid red line denotes the actual axes and object position. The green dashed line denotes the object after the region is defined and the landmarks are updated. The object is misshapen due to the low correlation in the base landmarks. (b) shows a LTR and an object defined inside for the case where the landmarks possess high correlation.



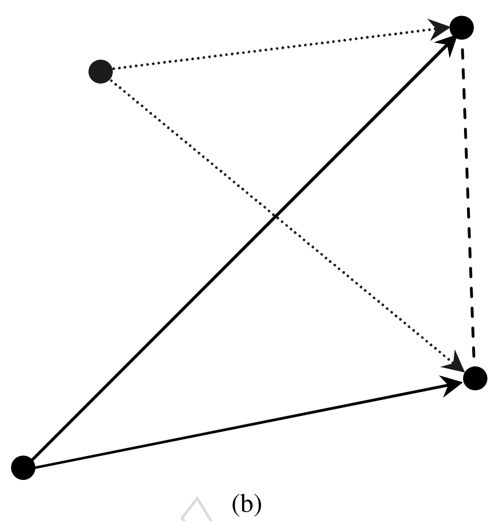


Fig. 9. Figure (a) shows a LTR which is not geographically well conditioned: the region could flip over. The dots represent the base landmark positions and the ellipses the uncertainty bounds. Figure (b) illustrates a case where two LTRs overlapped.

3.4.1. Virtual landmarks

There will be situations where part of the map cannot be included inside any local region. For example an environment with buildings. If the sensors cannot see through the building's walls and the robot cannot go inside the building, using the method presented, no regions could include the whole walls, since all the landmarks will be outside. In order to include the walls and keep the same type of regions throughout the map, the local regions could be formed with two real landmarks, and one *virtual landmark*, which could be created inside the building. The virtual landmark will have to be created when the vehicle is highly correlated with the other two landmarks. The difference between using two (the third landmark is artificial so does not have information) or three landmarks will be reflected in the accuracy of the dense map. The more landmarks used to define the axes of the local region, the less correlated will be the local map with the rest of the system.

Note that the only purpose of the virtual landmark is to have only one type of local region throughout the whole map, which then makes the implementation of the algorithm easier. However, a different strategy could be adopted, and in circumstances where a part of the environment is not enclosed by any LTR, a different type of region could be used. For example, three landmarks could be used to delimit the axes, and one axis could be extended to go inside the building. This strategy is more robust since three landmarks are used to create the local region but as mentioned before, this requires the algorithm to have different types of local regions and to have a mechanism to distinguish between the different scenarios to apply to each particular configuration.

3.5. Error Estimation in the Relative Represented Objects

It was mentioned in Section that the information fused in the local maps is decorrelated from the rest of the system. This was the main motivation behind the use of local representation for the dense maps; to reduce these correlations and then, even when they are not maintained, to make the approach a close approximation to an "optimal DenseSLAM". Neglecting these correlations does not introduce any inconsistency into the system, since only the landmark map is used for localization. The effect of the decorrelation will be an error in the position of the dense maps. This section derives the error introduced in the dense map resulting from neglecting the correlation with the landmark map.

Once the location of an environment property is fused inside the local regions, the algorithm never changes its local coordinates (α, β). A change in these coordinates can be produced by two means: direct observation of the property or through cross-correlation with the rest of the system. In this section we investigate the changes in the dense property position produced by cross-correlation with the rest of the system. Using the relative representation to define the dense maps results in a significant reduction in the correlation between the dense map locations and the rest of the system. Nevertheless, there will be some correlation due to the local vehicle uncertainty. The evaluation of this error is derived next.

After the vehicle takes new observations, the update produced to the first moment of the state vector is given by:

$$\hat{\mathbf{x}}^{+} = \hat{\mathbf{x}}^{-} + \Delta\mathbf{x} \tag{20}$$

where

$$\Delta \mathbf{x} = \mathbf{P}^{-} \nabla \mathbf{h_x} \mathbf{S}^{-1} \nu \tag{21}$$

Assume a range and bearing observation of the i-th landmark is taken

$$\mathbf{z} = \begin{bmatrix} h_r(\hat{\mathbf{x}}_{\mathbf{v}}^{-}, \hat{\mathbf{x}}_{\mathbf{Li}}^{-}) \\ h_\theta(\hat{\mathbf{x}}_{\mathbf{v}}^{-}, \hat{\mathbf{x}}_{\mathbf{Li}}^{-}) \end{bmatrix} \tag{22}$$

and the innovations vector ν and innovations covariance $\mathbf{S}$ are calculated:

$$\mathbf{S}^{-1} = \begin{bmatrix} S_1 & S_2 \\ S_3 & S_4 \end{bmatrix} \tag{23}$$

$$\nu = \begin{bmatrix} \nu_r \\ \nu_\theta \end{bmatrix} \tag{24}$$

Using eq. (21) it can be shown that the update produced in the mean of a non-observed landmark j due to the correlation with the vehicle and landmark i will be:

$$\begin{aligned} \Delta \mathbf{x}_j = {} & (\mathbf{P}_{jv} \nabla h_{r_\mathrm{v}}^T + \mathbf{P}_{ji} \nabla h_{r_\mathrm{i}}^T)(S_1 \nu_r + S_2 \nu_\theta) \\ & + (\mathbf{P}_{jv} \nabla h_{\theta_\mathrm{v}}^T + \mathbf{P}_{ji} \nabla h_{\theta_\mathrm{i}}^T)(S_3 \nu_r + S_4 \nu_\theta) \end{aligned} \tag{25}$$

where $\mathbf{P}_{jv}$ is the cross-covariance matrix between the landmark j and the vehicle pose. $\mathbf{P}_{ji}$ is the cross-covariance between the landmarks j and i, and:

$$\nabla h_{r_\mathrm{v}} = \frac{\partial h_r}{\partial \mathbf{x}_v} \tag{26}$$

$$\nabla h_{r_\mathrm{i}} = \frac{\partial h_r}{\partial \mathbf{x}_{Li}} \tag{27}$$

$$\nabla h_{\theta_\mathrm{v}} = \frac{\partial h_\theta}{\partial \mathbf{x}_v} \tag{28}$$

$$\nabla h_{\theta_\mathrm{i}} = \frac{\partial h_\theta}{\partial \mathbf{x}_{Li}} \tag{29}$$

Equation (25) represents the error in the location of the dense maps that the algorithm makes, by not considering the correlation between the dense information and the rest of the system. An example where eq. (25) is calculated for a local and for a global landmark is presented next.

EXAMPLE 2. The changes in position of a local represented landmark resulting from the correlation with the rest of the map are evaluated here, using the environment presented in Example 1. In Figure 10(a) the dashed line shows the evolution in time of the landmark i represented in local coordinates $\mathbf{x}_{Li}^{L}$, and the solid line illustrates the landmark position evolution when the landmark is expressed in global coordinates $\mathbf{x}_{Li}^{G}$. The lines depict the distance in meters of the landmark position with respect to the initial position estimated. The landmark is observed several times and once it is highly correlated with the base landmarks, it is not observed again. This is to imitate what actually occurs with the dense information, where there is no data association process. This means that once an observation is fused, it is assumed that the same point is never observed again. Since the landmark in this example is not directly observed, the changes in position are only due to the correlations with the rest of the map. Therefore the lines in Figure 10(a) are a representation of the error introduced by ignoring the update term shown in eq. (25), when the landmark is represented in local and in global coordinates. It can be observed that even when the position of the landmark in global coordinates changes more than 50 centimeters with respect to the initial position, its local coordinates only change approximately 6 centimeters when the loop is closed (largest update) and is rapidly stabilized to an almost constant value, while the global position continues changing. This 6 centimeters change in the local landmark coordinates is the error that the algorithm will make by not considering the correlations between the dense maps and the rest of the system, which as shown, is reduced drastically using the relative representation in comparison with the error produced using the global frame to represent the dense maps.

The points in Figure 10(b) depict the changes in position of the landmark i in global coordinates. The initial position is the bottommost point and then the estimate starts to move to where the actual landmark position is located.

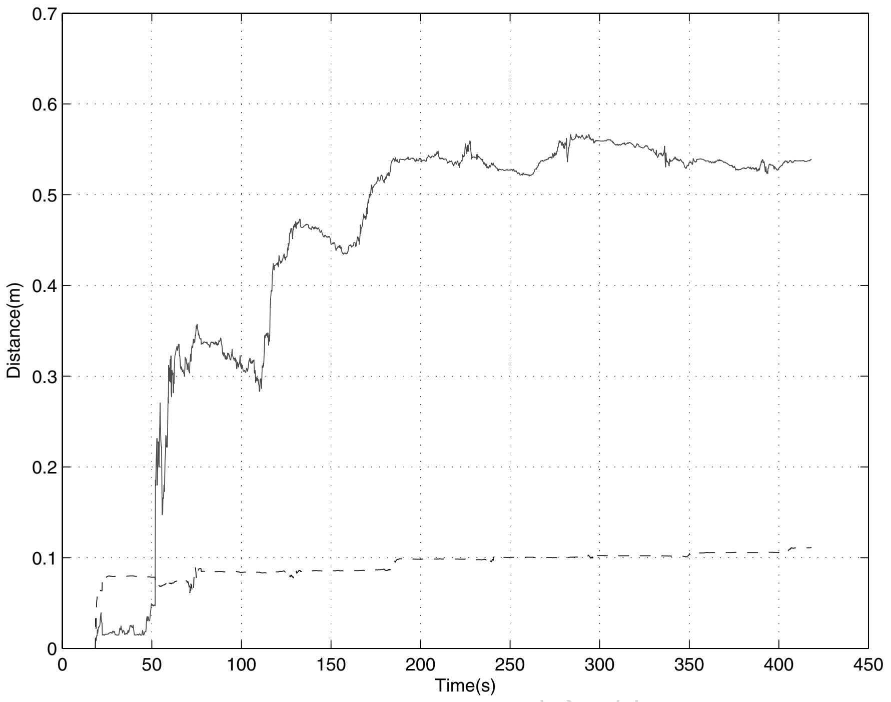
4. Consistency in DenseSLAM

This section discusses consistency in DenseSLAM. The next subsection will introduce a new concept named *Unidirectional Information Flow* UIF, which is used to demonstrate that the landmark map and the localization process obtained with DenseSLAM are optimal in comparison with feature-based SLAM.

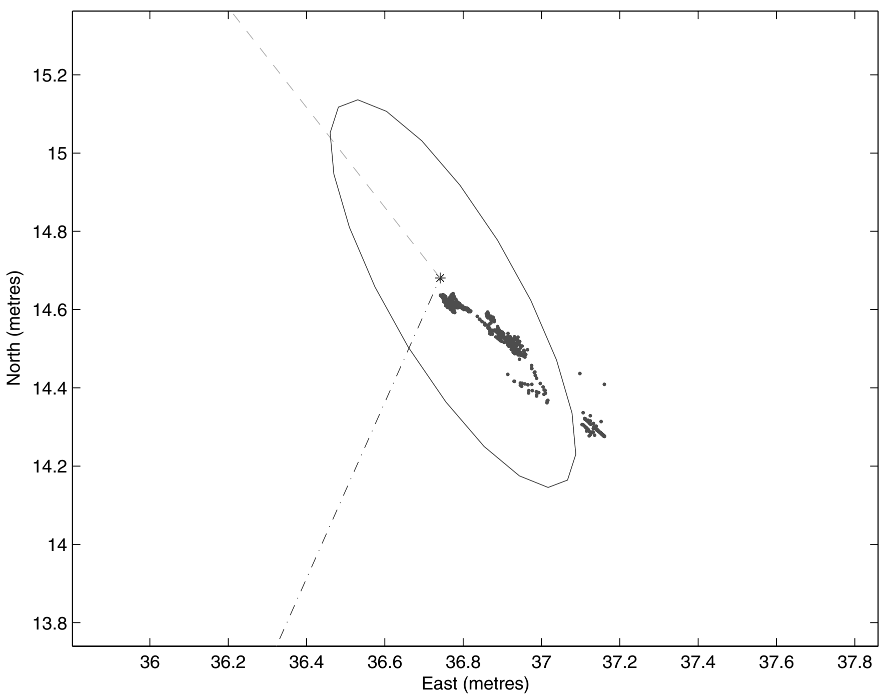
4.1. The Unidirectional Information Flow (UIF)

DenseSLAM is built upon two key variants of the basic SLAM algorithm.

- *The relative representation*. The representation stores the dense information in local coordinate frames reducing the cross-correlation between the dense representation and the landmark map.
- *The Unidirectional Information Flow*. The representation considers information going from the feature-based SLAM to the dense maps. There is no direct information flow from the dense maps to the feature-based SLAM. We have called this effect Unidirectional Information Flow (Nieto, Guivant, and Nebot 2004a). As a result of the UIF property, it is possible to augment the feature-based SLAM algorithm to obtain a



(a)



(b)

Fig. 10. Figure (a) shows the changes in the landmark's position, the solid line depicts the changes when the landmark is represented in global coordinates and the dashed line when it is represented in a global frame. The points in (b) depict the changes in position of the landmark *i* in global coordinates. The initial position is the bottommost point and then the estimate starts to move to the actual landmark position.

more detailed representation. The UIF guarantees the system remains consistent. This characteristic ensures that DenseSLAM will never perform worse than a standard feature-based SLAM.

The concept of *Unidirectional Information Flow* (UIF) for state-space estimation is now introduced, and the manner in which the HYMM representation uses this concept to ensure consistency in DenseSLAM is explained.

The notion of UIF is relevant to systems where the state-vector is divided into two groups: (i) states that both give to and receive information from the rest of the system and (ii) states that only receive information. These latter states are said to possess UIF. They are a set of states that receive indirectly the information obtained by the other states, but they do not give information back to the rest of the system.

Example 3 clarifies the idea, applying the UIF concept to the feature-based SLAM problem.[1]

EXAMPLE 3. Consider a landmark map where a subset of landmarks N_1 is estimated by the SLAM algorithm and where a second set of landmarks N_2 is observed only once. The latter set will not provide information to the rest of the system (because they are observed only once). However, because they are correlated to the set N_1 as a result of robot pose uncertainty they will receive information from N_1. As a consequence it is not necessary to maintain the cross-correlation within the landmarks set N_2, only the cross correlation between N_1 and N_2 will be necessary. A full filter implementation without considering UIF will require the update of a covariance matrix of $(N_1 + N_2)(N_1 + N_2)$. By considering the UIF property only $N_1(N_1 + N_2) + N_2$ elements of the covariance matrix have to be maintained. If $N_1 << N_2$, then the computational burden will be drastically reduced using the UIF concept.

The UIF is a general concept that can be used in different applications. Take for example an autonomous robot performing SLAM, and at the same time tracking people. Assume the motion model for the people is a random walk model. The state vector will consist of the robot pose, the static landmark positions and the states used to model the motion of the people, for example, speed and position. In general, there will be no information flow from the people's track to the robot pose, because the robot motion model will be more accurate than the random walk model used for the people tracks. However, the static landmarks have a noiseless model, so the robot will use the information coming from the landmarks to localize itself and its position for the tracking. This is a clear example where without adding restrictions to the system about how the information flows, the system naturally behaves under the UIF assumption. There is information flowing from the feature-based SLAM to the people tracks, but not information from the people tracks to the robot pose and the landmark position estimates. An application similar to this example was introduced in Wang (2004), where a vehicle does SLAM in the streets of a city, and at the same time the observations are used to track the cars moving around the vehicle. The implementation presented in Wang (2004) uses the information from the SLAM in the tracking algorithm, but it does not use the information from the tracking in the SLAM.

1. The example does not try to present a solution to the traditional feature-based SLAM problem, it is actually trying to illustrate the UIF concept in a very well known problem.

4.2. UIF in DenseSLAM

DenseSLAM uses the UIF to incorporate more information into the map representation. The algorithm maintains a joint state vector with the vehicle pose and feature positions and the dense maps are stored in a separate data structure.

This section demonstrates that by using the UIF concept, DenseSLAM can be decoupled into two problems: the classic feature-based SLAM and the estimation of the dense map.

Solving the DenseSLAM problem using a Bayesian formulation requires the evaluation of the following probability distribution:

$$\begin{aligned} &P(\mathbf{x}_k, \mathbf{f}, \mathbf{d}|\mathbf{Z}^k, \mathbf{U}^k, \mathbf{x}_0) \\ &\quad \propto \quad P(\mathbf{z}_k|\mathbf{x}_k, \mathbf{f}, \mathbf{d})P(\mathbf{x}_k, \mathbf{d}, \mathbf{f}|\mathbf{Z}^{k-1}, \mathbf{U}^k, \mathbf{x}_0) \end{aligned} \tag{30}$$

where $\mathbf{x}_k$ is the robot pose vector at time k, $\mathbf{f}$ is the features map, $\mathbf{d}$ is the dense map, $\mathbf{Z}^k$ is the set of observations received until time k, $\mathbf{z}_k$ are the observations at time k, $\mathbf{U}^k$ are the control inputs and x_0 the initial conditions.

Once the features are extracted, the observations can be divided into two groups; the observations belonging to the feature map and the ones belonging to the dense map.

$$P(\mathbf{z}_k|\mathbf{x}_k, \mathbf{f}, \mathbf{d}) = P(\mathbf{z}_k^f|\mathbf{x}_k, \mathbf{f})P(\mathbf{z}_k^d|\mathbf{x}_k, \mathbf{d}) \tag{31}$$

Now assume that the dense map does not provide any information to the robot pose and the feature map, then:

$$P(\mathbf{x}_k, \mathbf{f}|\mathbf{Z}^k, \mathbf{d}, \mathbf{U}^k) = P(\mathbf{x}_k, \mathbf{f}|\mathbf{Z}_f^k, \mathbf{U}^k) \tag{32}$$

This concept is illustrated in Figure 11. There is information going in both directions between the robot and the feature map, and information going from the robot and feature map to the dense map, but not information going directly from the dense map to the rest of the system. This is in accordance with what happens in practice during SLAM where geographically close objects present high correlation having high mutual information. If they share the same information, then only a selected part of the map needs to be used for the SLAM process (e.g., feature map). The rest of the information can be maintained and used for other tasks. For example, it can be used to maximize (or accumulate) all the information gathered from the sensors, e.g., to get an occupancy grid map (Nieto, Guivant, and Nebot 2004b), and then be used for the

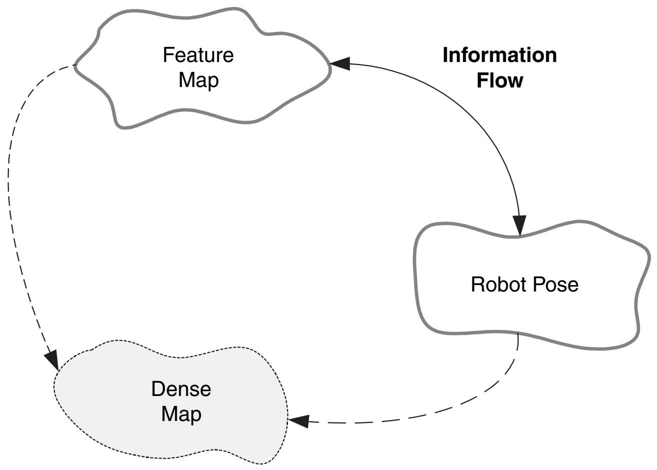


Fig. 11. *Unidirectional Information Flow* effect in DenseSLAM: The information between the features map and robot pose is bidirectional while the information between robot and dense map is unidirectional.

path planning algorithm. Applying eqs (31) and (32) to 30, the following result can be derived:

$$\begin{aligned} &P(\mathbf{x}_k, \mathbf{f}, \mathbf{d}|\mathbf{Z}^k, \mathbf{U}^k, \mathbf{x}_0) \\ &\propto \quad P(\mathbf{z}_k^f|\mathbf{x}_k, \mathbf{f})\, P(\mathbf{x}_k, \mathbf{f}|\mathbf{Z}_f^{k-1}, \mathbf{U}^k, \mathbf{x}_0) \\ &\qquad P(\mathbf{z}_k^d|\mathbf{x}_k, \mathbf{d})\, P(\mathbf{d}|\mathbf{Z}^{k-1}) \end{aligned} \tag{33}$$

The first factor on the right term of eq. (33) (second line) is a Bayesian feature SLAM, and the second factor (third line) corresponds to the dense Bayesian map estimation. This equation demonstrates that by virtue of the *Unidirectional Information* property the DenseSLAM problem can be decoupled into two problems: the estimation of the feature-based SLAM and the estimation of the dense map. This decoupling effect makes possible the implementation of an efficient algorithm able to obtain dense representations. Note that because of this decoupling effect, DenseSLAM can be implemented on top of any feature-based SLAM. This is one of the main characteristics of DenseSLAM, it is not restricted to a particular SLAM algorithm. For simplicity this paper presents implementations using a standard EKF-SLAM, but any of the efficient feature-based SLAM techniques (Knight, Davison, and Reid 2001; Williams 2001; Bailey 2002; Guivant 2002; Bosse et al. 2003) are equally valid.

5. Dense Maps Representation

One of the main strengths of DenseSLAM is the fact that it is not restricted to any particular local representation. The only condition required by the algorithm is to fuse the sensory information into a local representation whose frames are determined by the position of a set of landmarks.

This section shows two representations that can be used to represent the dense sensed information; DenseSLAM using Occupancy Grid (OG) and Sum of Gaussians (SOG) representation. These two representations with different characteristics were chosen in order to show the flexibility of the algorithm to represent the sensory information. Implementations of DenseSLAM using the two representations will be presented in the experimental section.

5.1. DenseSLAM Using Occupancy Grid (OG-DenseSLAM) Maps

The occupancy grid (OG) mapping technique (Elfes 1989b) represents the environment with a grid of fixed resolution. The OG is a multidimensional grid that maintains stochastic estimates of the occupancy state of each cell. As mentioned before, one of the main problems with OG maps is the lack of a method to include the correlations between cells. The technique assumes that the state variables of each cell are independent, an assumption that yields inconsistent results. This section addresses the problem of obtaining consistent OG maps. It will be shown that by using DenseSLAM it is possible to obtain grid maps, even in cases where the robot pose uncertainty is large.

DenseSLAM uses the landmark map to define the boundaries of a local grid map (see Figure 2). These local regions are then used as frames to fuse the dense information. In traditional OG maps, a grid of fixed resolution covering the region of interest is defined prior to starting the mapping process. DenseSLAM does not predefine the region of interest. The algorithm defines local OG maps at the time the robot explores new areas and incorporates more landmarks. In addition, different grid granularity can be assigned to each individual local OG map.

OG-DenseSLAM presents several advantages in comparison with classic OGs. First, OG-DenseSLAM overcomes the main problem of the classic OG framework: the local representation used by DenseSLAM allows the corrections to be propagated to the grid maps, which makes the final map consistent with the landmarks map and the robot localization process. In addition DenseSLAM naturally overcomes two other problems of traditional OGs: (i) DenseSLAM does not limit the grid to an area previously defined, instead local regions are created when new areas are explored and (ii) the algorithm permits regions to be defined with different resolutions, enabling the construction of maps with different levels of detail, depending on the importance of particular areas.

5.1.1. Observation Model

A stochastic sensor occupancy model is defined as:

$$P(\mathbf{r}|s(C_i) = Occ) \tag{34}$$

where **r** represents the sensor readings and $s(C_i)$ the state of the cell C_i which has two possible values; occupied or free.

The sensor occupancy model is obtained from the sensor model applying Kolmogorov's theorem (Elfes 1989a). Closed form solutions for eq. (34) can be derived for certain sensor models and numerical solutions are used in other cases. Different closed forms approximations for eq. (34) can be found in the literature (Pagac, Nebot, and Durrant-Whyte 1998; Ribo and Pinz 2001). The next example presents an occupancy model for the one-dimensional case.

EXAMPLE 4. Equation (35) illustrates an approximated closed form solution for the one-dimensional case (Leal 2003).

$$P(\mathbf{r}|s(C_i) = Occ) = \begin{cases} (1 - p_{max}) + (p_{max} - 0.5)(1 - \lambda) & -1 \le \lambda \le 0 \\ 0.5 + (p_{min} - 0.5)\lambda & 0 < \lambda \le 1 \wedge r \le r_u \\ 0.5 & r > r_u \end{cases} \quad (35)$$

where $\lambda = (1 - 2\lambda_r)$ and λ_r is the normalized sensor model; p_{min} and p_{max} correspond to the maximum and minimum of the occupancy model; and r_u corresponds to the closest range where:

$$0.5 + (p_{min} - 0.5)\lambda = 0.5 \quad (36)$$

See Appendix A in Leal (2003) for more details about this closed-form solution.

5.1.2. Update

OG approaches use Bayes formulation to fuse the sensor information collected.

$$P(s(C_i) = Occ|\mathbf{r}^{k+1}) = \frac{P(r_{k+1}|s(C_i) = Occ)P(s(C_i) = Occ|\mathbf{r}^k)}{\sum_{s(C_i)} P(r_{k+1}|s(C_i) = Occ)P(s(C_i) = Occ|\mathbf{r}^k)} \quad (37)$$

where $\mathbf{r}^{k+1}$ represents the set of observation vectors received until time k and $P(s(C_i) = Occ|\mathbf{r}^k)$ is the previous estimate of the cell state. It is important to note that the vehicle pose is assumed to be known in OG approaches, so the left term in eq. (37) should actually be written as $P(s(C_i) = Occ|\mathbf{r}^{k+1}, \mathbf{x}_v)$, where $\mathbf{x}_v$ represents the vehicle pose. To avoid repetition in the notation, $\mathbf{x}_v$ is not written here.

In practice the update is often solved using Odds formulation (Moravec 1988; Thrun 2002). The odds of a variable x with probability $P(x)$ is defined as $\frac{P(x)}{1-P(x)}$. The logarithmic form is used because it is computationally advantageous. The update using log Odds formulations is expressed as (for the sake of brevity, $s(C_i) = Occ$ is noted as C_i):

$$\log \frac{P(C_i|\mathbf{r}^{k+1})}{1 - P(C_i|\mathbf{r}^{k+1})} = \log \frac{P(C_i|\mathbf{r}_k)}{1 - P(C_i|\mathbf{r}_k)} + \log \frac{P(C_i|\mathbf{r}^k)}{1 - P(C_i|\mathbf{r}^k)} + \log \frac{1 - P(C_i)}{P(C_i)} \quad (38)$$

where $P(C_i)$ is the prior which is usually set to 0.5 and so the last term goes to zero. Note that eq. (38) does not need a normalization factor as in eq. (37).

5.2. DenseSLAM Using Sum of Gaussians (SOG-DenseSLAM)

This section presents a SOG-DenseSLAM algorithm. A SOG distribution is used to represent the sensory information. This SOG is then fused into the map using local representation in the same form as the OGs in the previous section. It is important to clarify that the final SOG map does not represent a distribution, therefore cannot be used for recursive mapping. However, the 2D PDF representation can be used, for example to assist the data association process via scan correlation or for localization after a final map has been obtained with SOG-DenseSLAM.

5.2.1. Observation Model

An n-dimensional Gaussian distribution is defined as

$$g(\mathbf{x}; \bar{\mathbf{x}}, \mathbf{P}) \triangleq \frac{1}{\sqrt{(2\pi)^n |\mathbf{P}|}} \exp\left(-\frac{1}{2}(\mathbf{x} - \bar{\mathbf{x}})^T \mathbf{P}^{-1}(\mathbf{x} - \bar{\mathbf{x}})\right) \quad (39)$$

where $\bar{\mathbf{x}}$ and $\mathbf{P}$ are the mean and covariance, respectively. An n-dimensional sum of Gaussians (SOG) is defined as the sum of k scaled Gaussians.

$$G(x) \triangleq \sum_{i=1}^{k} \alpha_i g(\mathbf{x}; \bar{\mathbf{x}}_i, \mathbf{P}_i) \quad (40)$$

where, for a *normalized* SOG, the sum of the scaling factors α_i is one.

Any set of point measurements with Gaussian noise can be represented as a SOG. In particular, a range and bearing measurement in polar coordinates $\mathbf{z}_i = (r_i, \theta_i)$ with covariance matrix R, can be represented in cartesian coordinates as

$$\mathbf{x}_i = \mathbf{f}(\mathbf{z}_i) = \begin{bmatrix} r_i \cos\theta_i \\ r_i \sin\theta_i \end{bmatrix} \quad (41)$$

with covariance matrix

$$\mathbf{P}_i = \nabla\mathbf{f}_{\mathbf{z}_i} \mathbf{R}_i \nabla\mathbf{f}_{\mathbf{z}_i}^T \quad (42)$$

where $\nabla\mathbf{f}_{\mathbf{z}_i} = \frac{\partial\mathbf{f}}{\partial\mathbf{z}_i}$.

Figure 12 shows an example of a laser scan represented as a SOG. In this example, the SOG representation was built with Gaussians of equal height, which means that the scaling factors in eq. (40) are equal to $\sqrt{(2\pi)^n |\mathbf{P}|}$, where $\mathbf{P}$ is the Gaussian covariance matrix.

5.2.2. Update

To understand the update it is necessary, first, to clarify what the Gaussians are actually representing. The Gaussians represent the probability of the area under the Gaussian shape being occupied. But they do not formally represent a probability distribution, thus the representation cannot be used for recursive data fusion. However, the Gaussians give a good representation of the shape of the environment. The strategy for the update will be to cover the areas where observations are received with the smallest possible number of Gaussians. This could be done for example by fusing two Gaussians that are close in only one Gaussian with mean equal to the average of the means and variance equal to the sum of the variances.

The algorithm implemented here works as follows:

1. When an observation frame is received, it is represented as a SOG as shown before.
2. To fuse the new SOG, a nearest neighbor algorithm associates each Gaussian from the *a priori* map that is inside the sensor view, with a Gaussian from the observation SOG.
3. A d_{min} value is defined, so if two Gaussians from the *a priori* map and the observation scan are associated and the distance between the Gaussians mean is smaller than d_{min} then the two Gaussians will be fused in one new Gaussian.
4. The Gaussian's fusion consists of obtaining a new Gaussian from the two associated Gaussians. This is done as follows. The mean of the new Gaussian $\mathbf{x}_{new}$ is equal to the average of the means.

$$\mathbf{x}_{new} = \frac{\mathbf{x}_{priori} + \mathbf{x}_{obs}}{2} \tag{43}$$

To obtain the new Gaussian variance σ_{new}, the variances of the two associated Gaussians are first analyzed.

If the sum of the Gaussian variance from the *a priori* map and the Gaussian variance from the observation is smaller than $k \cdot d_{min}$:

$$\sigma_{priori} + \sigma_{obs} \leq k \cdot d_{min} \tag{44}$$

then the covariance matrix of the new Gaussian is equal to the sum of the covariances.

$$\mathbf{P}_{new} = \mathbf{P}_{priori} + \mathbf{P}_{obs} \tag{45}$$

If the sum of the Gaussian variance from the *a priori* map and the Gaussian variance from the observation is larger than $k \cdot d_{min}$ then the covariance matrix of the new Gaussian is equal to the covariance matrix of the *a priori* Gaussian:

$$\mathbf{P}_{new} = \mathbf{P}_{priori} \tag{46}$$

This last point is to ensure that the Gaussians' variance will not continue growing when more observations are received, so if the sum of the Gaussians is large in comparison with the distance between them, then the covariance of the new Gaussian stays as the one from the *a priori* Gaussian.

Experimental results of OG-DenseSLAM and SOG-DenseSLAM will be presented in Section 7.

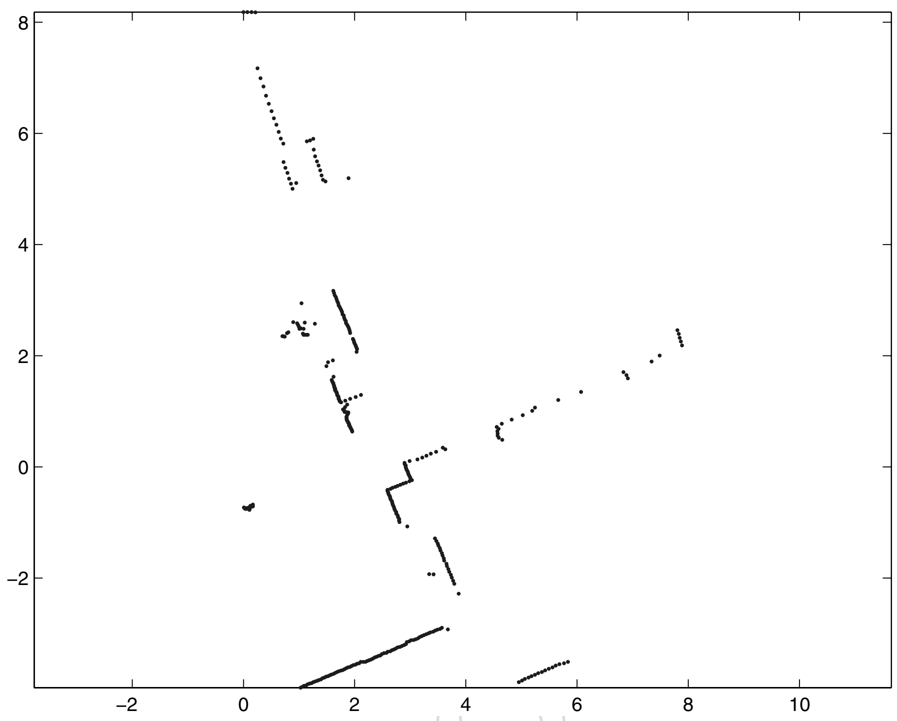
6. DenseSLAM: Applications

This section shows how the detailed multi-dimensional environment description obtained by DenseSLAM can be used to improve the vehicle navigation process. Two particular applications are shown. (i) Complex landmarks can be extracted and incorporated as they become identified using the dense representation. The incorporation of landmarks into the navigation map will improve the vehicle localization process. (ii) The dense maps can be used to estimate the variations in time of the areas explored by the robot which can be used to discriminate whether a region has potential dynamic objects. This information will help to prevent the vehicle from using dynamic objects for the localization process.

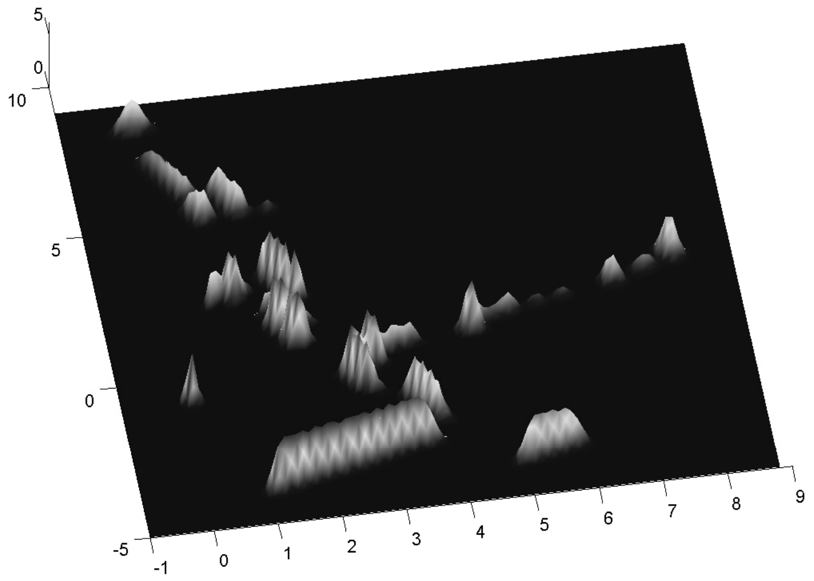
6.1. High Level Landmarks (HLLs)

One of the main problems in SLAM algorithms is the error accumulation resulting from non-linearities in the system. This error accumulation can be reduced if more information is added into the localization map, because the vehicle error will remain smaller. Among the reasons to avoid including more landmarks is the computational burden required to maintain the map. However, in many situations, even when the computational cost may not be a problem, the difficulties of finding *stable and easily detectable* features cause the algorithm to use only a small number of landmarks for the localization process, which results in a major accumulation of errors resulting from non-linearities.

DenseSLAM yields a rich environment representation, which gives the possibility of adding landmarks extracted from the dense maps into the landmarks map. In many situations an object cannot be detected using the measurements taken from only one vantage point. This can be for a variety of reasons: occlusion between objects, the size of the object in relation to the sensor field of view, an inappropriate feature model, or just because the nature of the sensor makes



(a)



(b)

Fig. 12. Figure (a) shows the set of raw range-laser data points transformed to a sensor-centric coordinate frame. (b) Shows the SOG representation of this scan.

the estimation of the landmark location impossible from only one vantage point (e.g., wide-beam sonar: McKerrow 1993; Leonard et al. 2002). Estimating partially observable features has been an important research topic in computer vision using stereo vision and bearing only information, where the initialization of the feature position is a significant problem. The problem of partially observable features has also been studied for localization and SLAM applications. In Leonard et al. (2002) an approach is presented that delays the decision to incorporate the observations as map landmarks. Consistent estimation is achieved by adding the past vehicle positions to the state vector and combining the observations from multiple points of view until there is enough information to validate a feature. In McKerrow (1993), intersection of a range of constant depth of ultrasonic sensors is used to determine the location of features from multiple vantage points.

Having a comprehensive representation of the environment will enable a delayed processing to determine whether part of the map can qualify as a landmark. The rich representation obtained by DenseSLAM will enable postprocessing capabilities to continuously detect high-level landmarks using the dense map layers. The newly detected landmarks can then be added to the feature map. This approach has the potential of incorporating a large number of landmark models, some of them to be applied online at the time the observations are taken and the rest to run in the background when computer resources become available. The landmarks can then be incorporated into the features map.

6.1.1. HLLs Initialization

When a HLL is detected, it has to be incorporated in the state vector in order to be used for SLAM. The HLLs cannot be initialized directly with the information extracted from the dense maps since this will violate the UIF principle. Thus, once the HLL is detected it will remain as a passive landmark until the vehicle actually observes it again. At that moment, the HLL will be initialized using the new observations.

6.1.2. HLLs Representation

The only condition for the incorporation of a HLL is to represent the information in the same form as the feature map. In the implementations presented in this paper EKF-SLAM is used, thus the HLLs have to be represented in state vector form.

The HLLs could be represented using geometric parameters. Experimental results of SLAM using trees as landmarks are presented in Guivant (2002). An EKF is run which estimates the trees' parameters, which consist of the center and the diameter of the trees' trunks. In Thrun et al. (2001) an algorithm is presented that employs expectation maximization to fit a low-complexity planar model to 3D data collected by range finders and a panoramic camera. After this model is obtained, its parameters could be added to the state vector to represent a HLL.

In this paper we opted for representing the HLLs as a local coordinate system and a template which is defined relative to the local axes. The templates are formed with the information extracted from the dense maps. Scan correlation is used in order to generate observations of the landmarks (see Nieto, Bailey, and Nebot 2005 for more details). An example of this representation is shown in Figure 13. Figure 13(a) shows a HLL extracted from the OG map layer obtained by DenseSLAM. The figure shows the landmark template and the local coordinate system. Figure 13(b) shows the same HLL but now extracted from the SOG layer.

6.2. Dynamic Environments

Most of the mapping algorithms assume the world is static (Thrun 2002). Dynamic environments require an extension of the typical representation used for static environments. That extension should allow for modeling the temporal evolution of the environment. Dynamic objects can induce serious errors in the robot localization process. Only a few approaches that include moving objects have been presented so far. The next paragraphs review some of them.

A SLAM algorithm with generic objects (static and dynamic) is presented in Wang (2004). Similar to classic SLAM, the approach calculates the joint posterior over robot and object's pose, but unlike traditional SLAM it includes also the object's motion model. The problem is shown to be computationally intractable and so a simplified version called *SLAM with Detection and Tracking of Moving Objects* (SLAM with DATMO; Wang, Thorpe, and Thrun 2003) is presented. The latest algorithm decomposes the estimation process into two separate problems: (i) the SLAM problem, using static landmarks as the classic approach, and (ii) the detection and tracking of moving objects, using the robot pose estimated by the SLAM algorithm. This simplification makes updating both the SLAM and the tracking algorithm possible in real-time since they are now considered to be two independent filters.

In Hähnel et al. (2003) an algorithm for mapping in dynamic environments is presented. The aim of the approach is to determine which measurements correspond to dynamic objects and then filter them out for the mapping process. The approach uses the EM algorithm; the expectation step computes an estimate of which measurements might correspond to static objects. These estimates are then used in the maximization step to determine the position of the robot and the map.

An approach called *Robot Object Mapping Algorithm* (ROMA) is presented in Biswas et al. (2003). The main goal is to identify non-stationary objects and model their time varying locations. The approach assumes that objects move sufficiently slowly that they can safely be assumed static for the time it takes to build an occupancy grid map of the whole

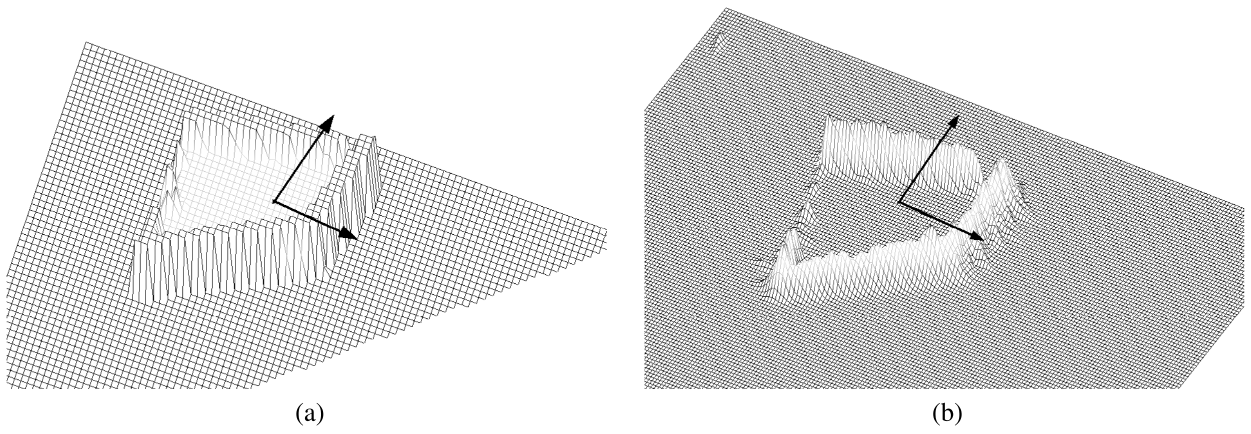


Fig. 13. The figure shows a HLL extracted from the dense maps. (a) Shows the HLL represented using the OG map and (b) the SOG map.

area explored by the robot. Assuming the robot is able to acquire static occupancy grid maps at different times, changes in the environment are detected using a differencing technique. The algorithm learns models of the objects using EM. The expectation step calculates the correspondences between objects at different points in time and the maximization step uses these correspondences to generate refined object models, represented by occupancy grid maps.

The algorithms presented in Wang, Thorpe, and Thrun (2003) and Montemerlo, Thrun, and Whittaker (2002) have one thing in common; they rely on pre-defined models of the specific objects they aim to track. ROMA, however, is able to learn about the shape of the objects, but the algorithm presents a number of limitations. Objects have to move slowly (it is not able to cope with fast-moving objects such as people); it is assumed the robot is able to obtain static maps at different times. The results presented include only four different objects in an environment where these objects can be perfectly segmented from a laser scan. The extension to a real environment with a larger number of objects may not be possible and will be computationally very expensive.

If navigation is the primary objective, the accurate shape of objects, or even their classification may not be important in general. What may be more useful is an algorithm able to identify observations that may be coming from objects that are not static and eliminate them from the list of observations to be used for the navigation process. Furthermore the algorithm could identify areas where it is more likely to find dynamic objects (e.g., a corridor where people walk) and then avoid their use or give a low priority to observations coming from objects in those areas.

The rich environment representation obtained by DenseSLAM allows a map layer identifying the most likely areas to possess dynamic objects to be built. As shown in Biswas et al. (2003), dynamic objects can be identified by differentiation of maps taken at different times. There are two different classes of object motions in a dynamic environment; slow motion, as for example the motion of a bin, which will be static during most of the day but will eventually be moved; and fast motion, such as people. The next subsections show how, using DenseSLAM, and applying a straightforward differentiation, it is possible to identify regions with dynamic objects for either fast or slow motion objects.

6.2.1. *Slow Motion*

One of the main problems with the differentiation is that maps obtained at different times will have different uncertainty. If a global map is maintained and two maps acquired at different times need to be differentiated, the uncertainty in the maps will make the matching process very difficult.

In DenseSLAM the global map is divided into smaller regions, so the whole map can be differentiated by applying differentiation between the corresponding local regions. As a consequence, the differentiation process will be prone only to local errors (which as shown in Section 3.5 are much smaller than the global ones) eliminating detection errors due to the uncertainty between maps acquired at different moments.

A particular case where the DenseSLAM representation will not present advantages over other approaches is in multi-robot mapping. If multiple robots are used to map an area, each robot will form the map using different local regions. If the objective is to detect dynamic objects by fusing the maps built by different robots, a global map will have to be used and DenseSLAM loses advantages with respect to other approaches.

6.2.2. *Fast Motion*

Fast motion can be captured by differentiation, in a similar way to slow motion. The main difference is that the differ-

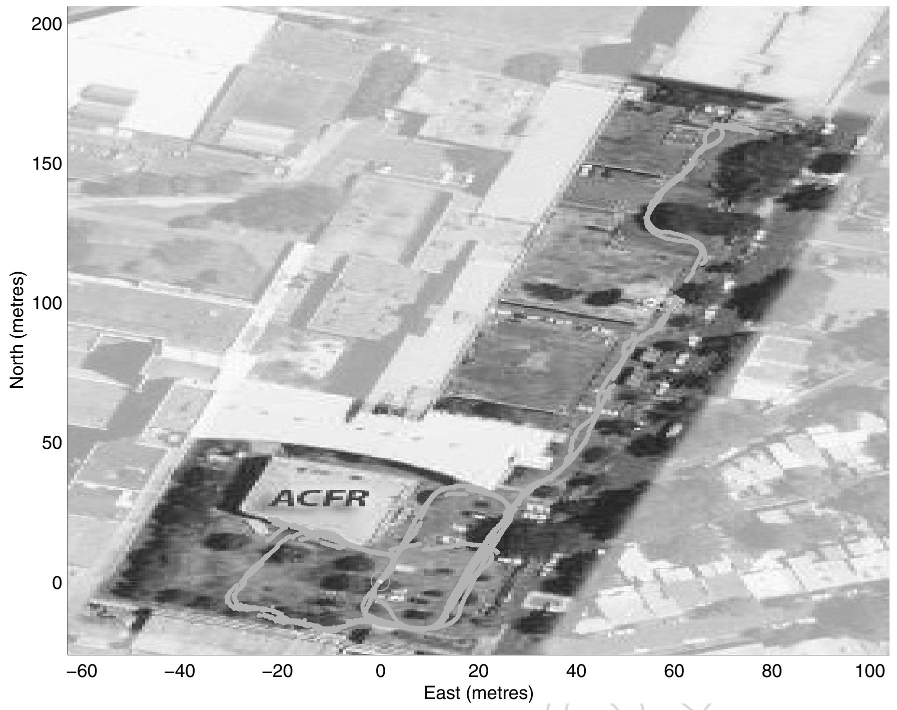


Fig. 14. Satellite picture of the experimental environment with the trajectory reported from the GPS.

entiation is done over shorter periods of time and only in the region under the sensor view. As in the detection of objects with slow motion, using DenseSLAM the differentiation is done using local regions instead of the global map. The motion detection will be included in a map layer as will the other properties captured by the sensors, then the global position of the dynamic map will be updated together with the other map properties (color, occupancy, etc.).

It is important to note that fast motion detection can be also done with other techniques that are able to create a dense map in proximity to the vehicle position. The advantage of DenseSLAM is that the dense representation is already obtained, therefore, the detection of moving objects is a straightforward procedure that does not add computational cost.

7. Experiments

This section presents experimental results in an outdoor environment. Figure 14 shows a satellite picture of the environment used for the experimental test. The environment is a large area of 120 by 200 metres and the run is approximately 1 km long.

The experimental platform used for the experiments is a conventional Holden UTE (see Extension 4). The platform is equipped with Sick lasers, a linear variable differential transformer sensor for the steering mechanism, back wheel velocity encoder, inertial unit, compass and GPS. The inertial unit and wheel encoder were used for the vehicle dead-reckoning process, and a Sick laser was used to take external observations. The dataset used for the experimental results and movies of the experiment can be downloaded at Guivant and Nebot (2002a). See also Extension 2.

7.1. DenseSLAM

Figure 15(a) presents the results obtained with the dead reckoning sensors. The figure shows the path obtained with the motion model and the GPS information. Because of the nature of the environment (buildings and trees), GPS was not always available. However, there were still some sections of the path where the GPS was available with good quality and could be used as a reference. Figure 15(b) shows the result obtained with the classic EKF-SLAM. The landmark mean and the 3σ ellipses covariance bounds are also shown. The vehicle starts to move at $(0, 0)$ coordinates. The vehicle initially completes a couple of loops in the car park area near the ACFR building and then it moves to the top part of the run. This is the part that presents the largest uncertainty. A zoom-in of the top part is shown in Figure 16.

In order to test the DenseSLAM algorithm, the GPS information was fused with the feature-based SLAM to obtain a laser image of the environment that is used as a reference to compare with the estimates by DenseSLAM. Figure 17 shows the laser image obtained with the GPS information

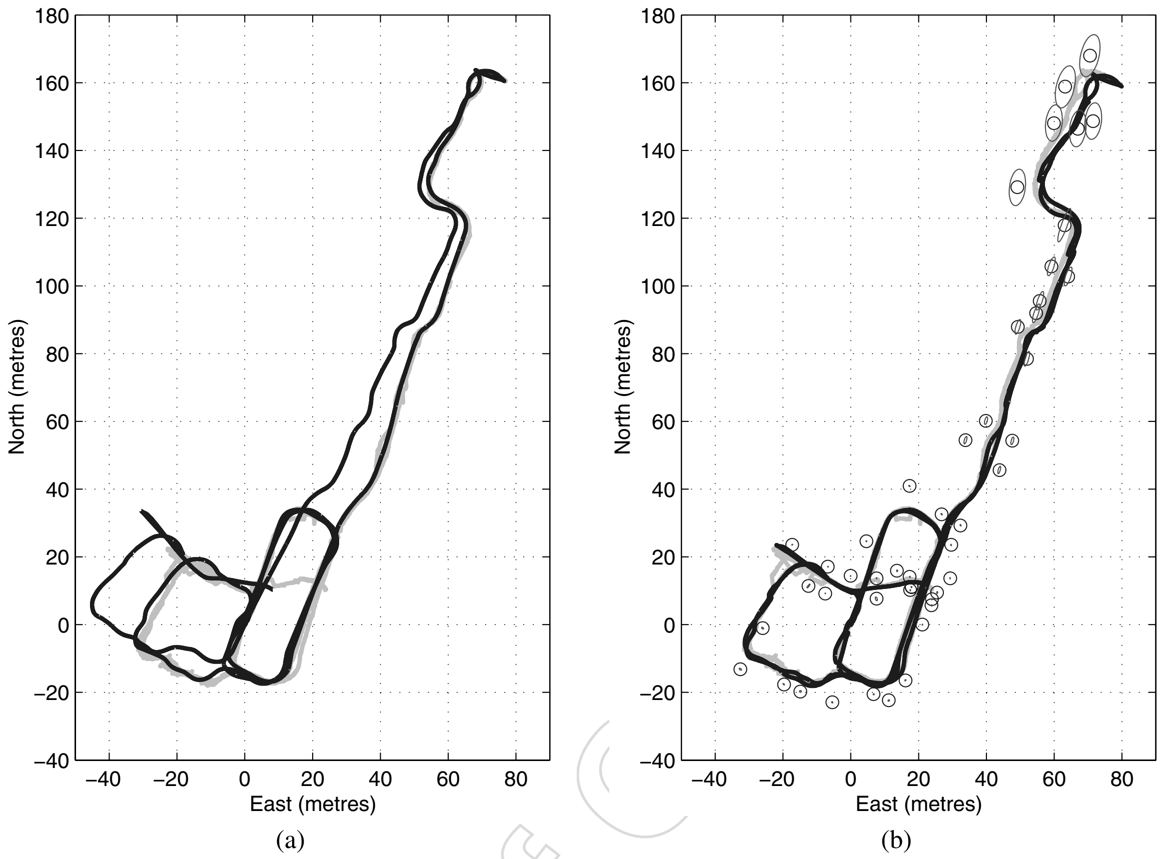


Fig. 15. The dark line in (a) shows the path obtained with the dead-reckoning sensors and the position reported by the GPS is depicted by the light line. (b) Shows the result obtained with the SLAM algorithm. The ellipses represent the 3σ bound for the landmarks uncertainty.

and the final map obtained with DenseSLAM. The dense map was obtained by fusing the raw laser observations into the local regions. The figure also shows the landmark positions. Because of the large number of buildings present in the environment, virtual landmarks (see Section 3.4.1) were needed to incorporate the walls into the map. This is why some extra landmarks can be observed in Figure 17 with respect to the feature-based SLAM result.

Figure 18 shows a zoom of the top part of the run. Figure Figure 18(a) shows the result obtained by DenseSLAM before closing the loop. The figure also shows the laser image used as a reference. The error in the estimated map before closing the loop can be easily observed. Figure 18(b) shows the result after closing the first loop. Although there is still some residual error, it is clear how the estimated map has been corrected. Looking at Figure 15(b) it can be seen that there is some remaining uncertainty in these landmarks even after the loop is closed. This is because the vehicle does not return to the top part of the run after closing the first loop. As a result, the error in that region is not reduced as much as in the bottom part of the run. Nevertheless, an important correction in all the regions has still been made.

7.2. Multi-Layer Maps

To illustrate the multi-layer capabilities of DenseSLAM, an implementation of the algorithm which obtains different representations using the laser information was carried out.

The experimental results are obtained using a part of the run presented before. The use of a smaller area provides a clearer illustration of the multi-layer concept. Figure 19 shows the section of the run utilized. The green solid line depicts the vehicle trajectory as reported by the GPS. The blue dots represent a laser image obtained using SLAM in combination with GPS.

Three different map-layers were acquired using the laser information. The first layer was obtained by fusing the raw observations. The second layer was obtained by implement-

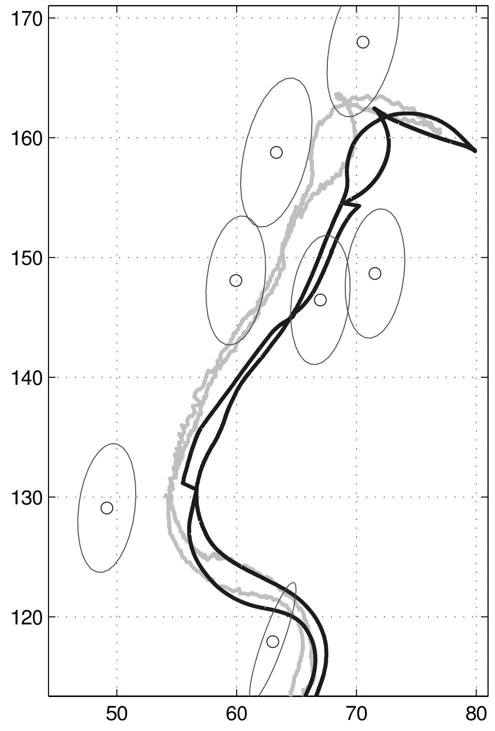


Fig. 16. Zoom in of the top part of the environment. The figure presents the result obtained with the feature-based SLAM.

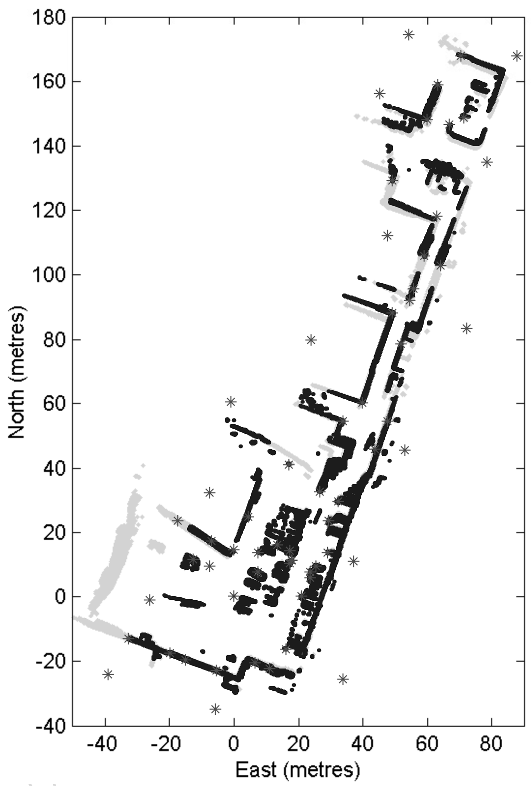


Fig. 17. Final map obtained with DenseSLAM. The light points represent the laser image obtained using GPS and SLAM. The dark points depict the dense map estimated by DenseSLAM.

ing OG maps inside the local regions. And the third layer consists of a Sum of Gaussians (SOG) representation of the environment.

Figure 20 shows the map obtained fusing the raw observations in the local regions. Figure 21 illustrates the occupancy map layer obtained by DenseSLAM. The grid resolution is 0.5 meters. Figure 22 shows the SOG map layer acquired. The figure shows the ellipses which represent the 1σ bounds for the Gaussians covariance. The SOG map was obtained using a $d_{min} = 0.5$ meters for the nearest neighbor algorithm. Figure 23 illustrates the SOG map result. The SOG surface was built using a grid of 0.5 meters. Finally, Figure 24 illustrates the three map layers obtained, from bottom to top, the figure shows the raw data map, the SOG and the OG layer.

7.3. Applications

7.3.1. High Level Landmarks (HLL)

An algorithm to detect high level landmarks (HLLs) was run using the representations obtained with DenseSLAM. The implementation creates two map layers, a SOG and a OG layer.

Since the environment is dominated by buildings, the HLLs were defined as corners. To make the HLLs more "unique", in the implementation presented the algorithm searches for pairs of corners (two corners in proximity). Thus, for a part of the map to qualify as a HLL, it needs to have two corners and the corners need to be closer than a predefined distance. A template area is delimited around the two corners, and if the SOG map-layer under the template area possesses more than a minimum number of Gaussians, the template is accepted as a HLL.

The first step is then to search for corners, this was done using the OG map layer. Figure 25 shows the result of the corner detector algorithm. A Harris corner detector was implemented and run in the OG layer. The crosses represent the position of the corners identified. After the corners are detected, the algorithm searches for pairs of corners whose distance apart is less than 7 meters. Then a rectangle area around the two corners is delimited and the HLL is extracted from the SOG taking the Gaussians under the rectangle. A HLL is accepted only if it possesses a minimum number of Gaussians.

In this implementation the HLLs' templates are the Gaussians extracted from the SOG-map layer. Figure 26(a) illustrates the HLLs found. The figure shows the laser image, the corners detected and the HLL bounds are represented by the

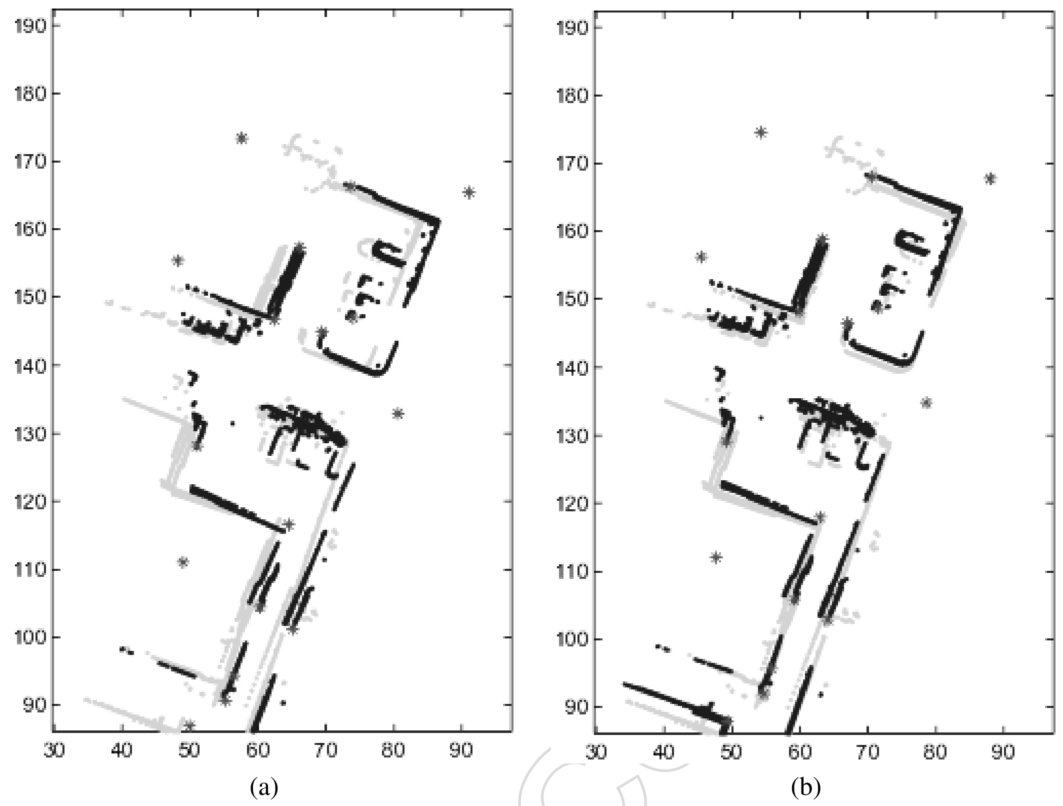


Fig. 18. The lighter points represent the laser image obtained using GPS/SLAM. The darker points represent the final map obtained with DenseSLAM. Figure (a) shows the result before closing the loop, and (b) after the loop is closed.

rectangles. Nine HLLs were accepted by the algorithm. Figure 26(b) shows the HLLs map. The points represent the mean value of the Gaussians that form the templates for the landmarks. After the HLLs are detected, they are incorporated in the state vector. The landmarks extracted from the dense map cannot be immediately initialized because that will violate the UIF condition, which will introduce inconsistencies into the system. Then, the HLLs are incorporated in a list of passive landmarks. These passive landmarks will be added into the state vector the next time the vehicle observes them again. For the implementation presented here, SOG-correlation is used in order to generate observations of the SOG-templates (Bailey 2002; Nieto, Bailey, and Nebot 2005).

7.3.2. Dynamic Maps

The slow motion detection algorithm was also tested. Figure 27(a) and (b) show images of OG map layers obtained in the car park area at different times. The map shown in Figure 27(b) was obtained after three cars were removed from the car park area.

The detection of dynamic objects is done by a function running in the background which differentiates maps obtained at different times. Not only will the dynamic objects be captured by the differentiation, but also the borders of the static objects resulting from sensor noise. To eliminate this effect, a Gaussian lowpass filter is run after the differentiation which removes the contours of the static objects. After this, a threshold operation is done, and then cells with difference in occupancy greater than 0.5 are set as dynamics, and cells with difference less than 0.5 remain as statics. After applying the thresholds the result is a binary map with values 1 for the dynamic cells and 0 for the static cells. The dynamic map layer can be fused with the *a priori* dynamic map simply by applying a logical OR operation.

The slow motion detection was run using the maps shown in Figure 27(a) and (b). The grid cells detected as dynamics are depicted with lighter color in Figure 27(a). As can be seen, variations have been identified in two different places. The first place on the right is where two cars were parked, and the second spot is where a third car was parked.

The main objective pursued here to identify dynamic regions is to avoid the use of possible landmarks formed by dynamic objects. Then, after the algorithm detects dynamic areas, the result is superimposed with the navigation map and the landmarks that are under one of the dynamic areas are

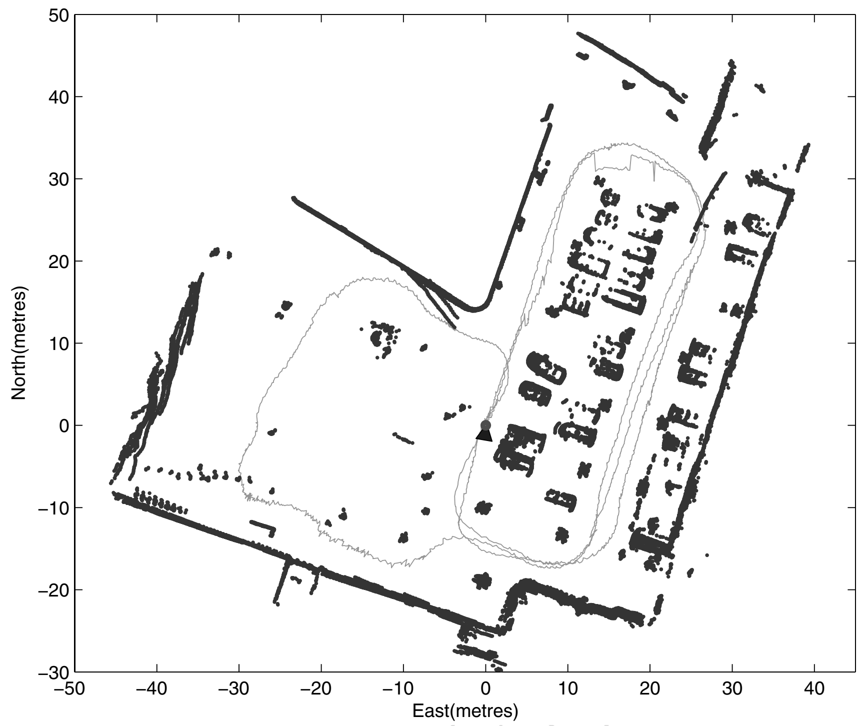


Fig. 19. Experimental environment. The green solid line depicts the vehicle trajectory as reported by the GPS. The blue dots represent a laser image obtained using SLAM in combination with GPS.

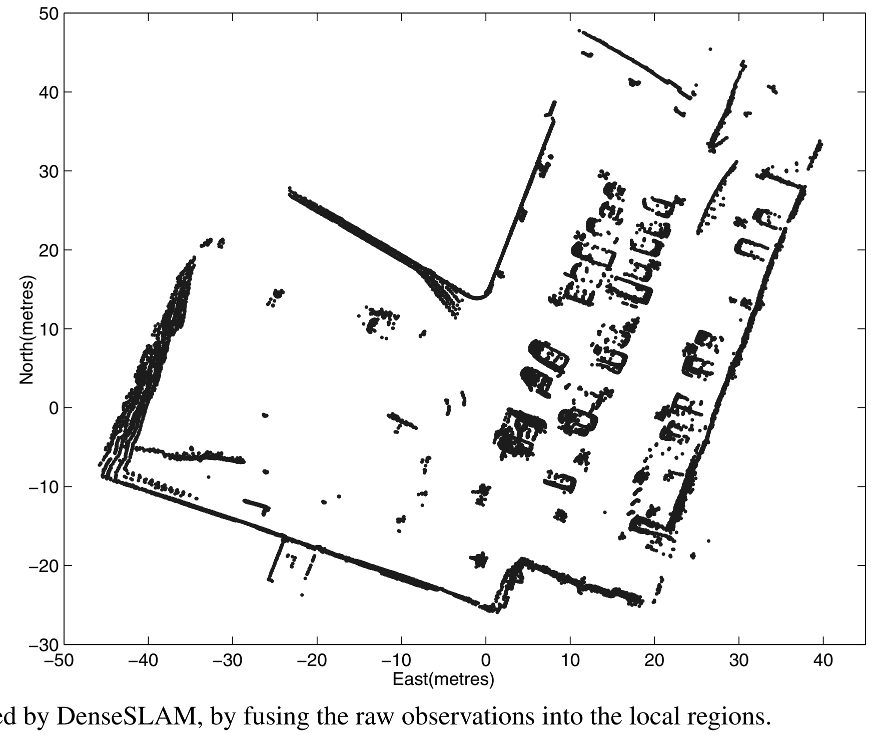


Fig. 20. Map obtained by DenseSLAM, by fusing the raw observations into the local regions.

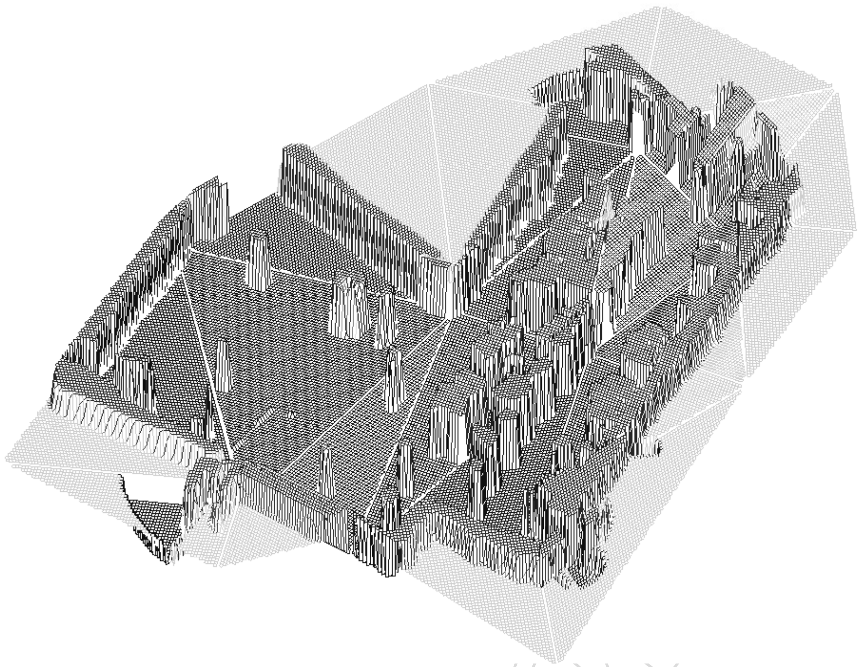


Fig. 21. The surface represents the OG map obtained with OG-DenseSLAM.

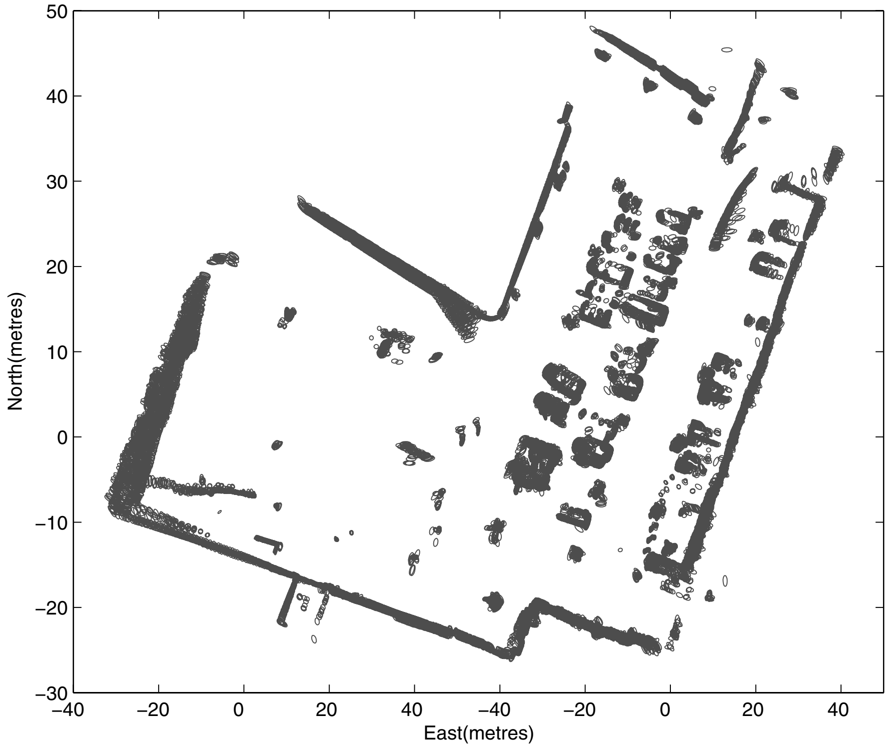


Fig. 22. SOG map layer. The ellipses represent the 1σ bounds for the Gaussians covariance.

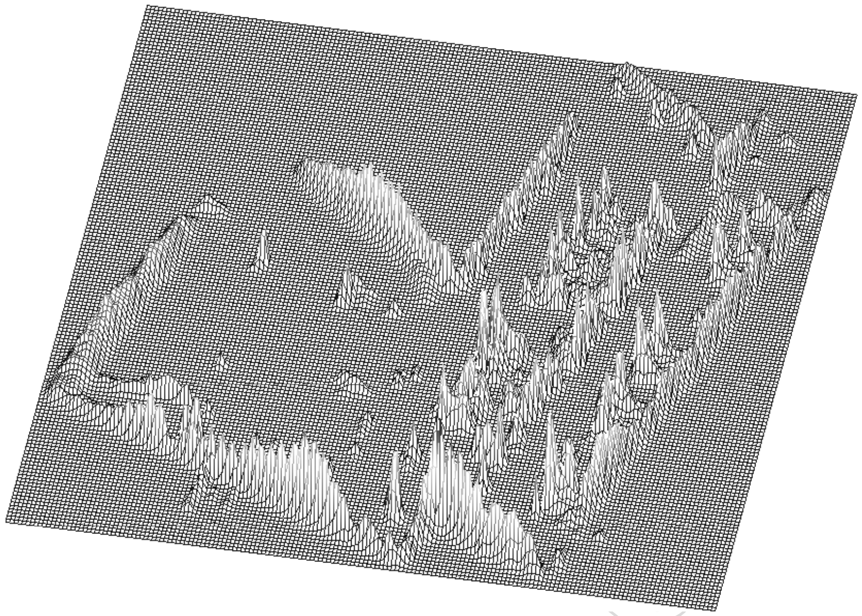


Fig. 23. The surface represents the SOG map obtained with SOG-DenseSLAM.

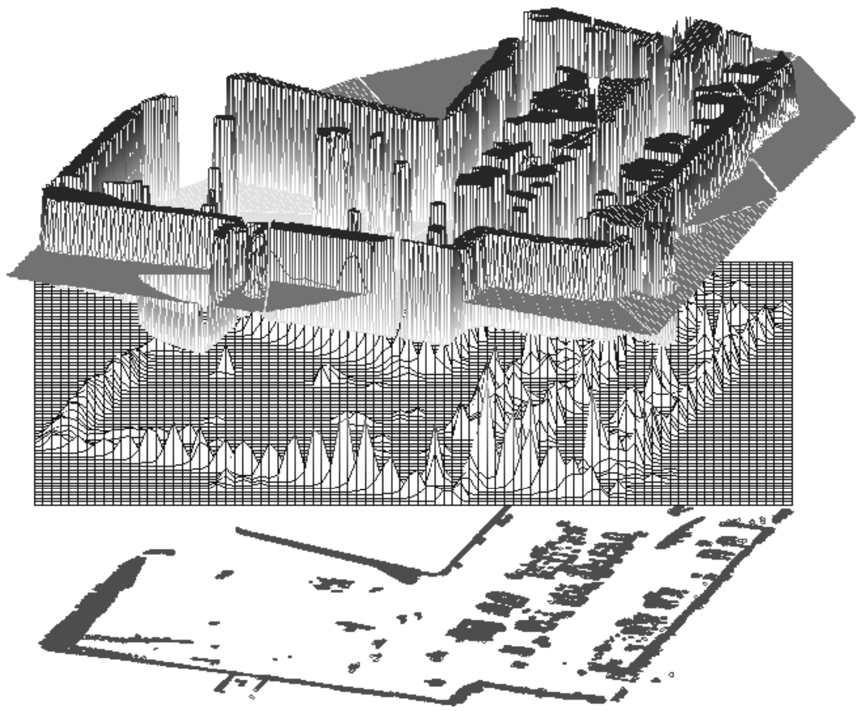


Fig. 24. Three map layers obtained with the laser information. From bottom to top, the raw data, the SOG and the OG layers.

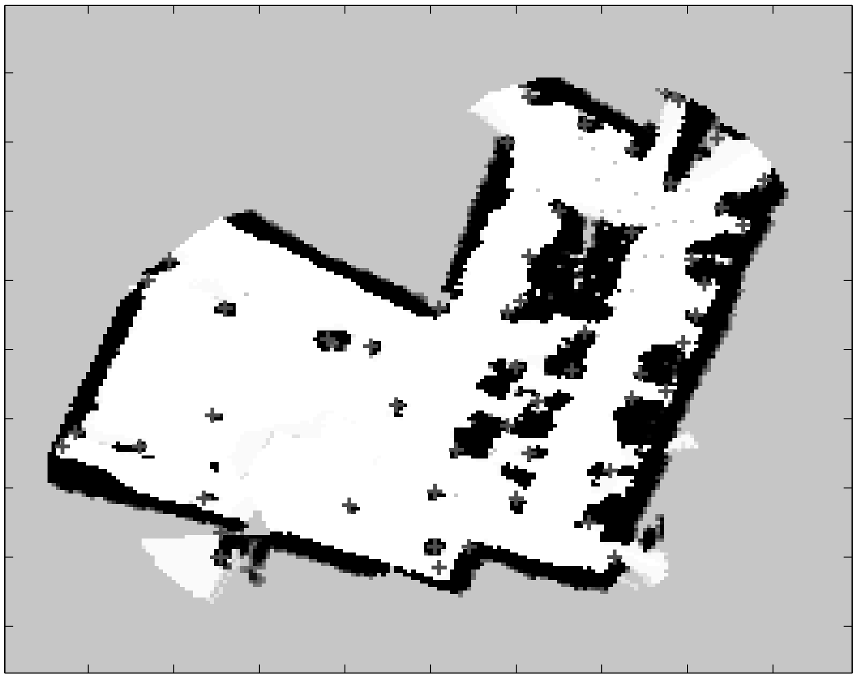


Fig. 25. OG map layer with corners detected.

removed from the state vector. Figure 28 shows the HLLs map and the dynamic areas detected which are denoted by lighter points. The HLL represented with a dashed line is clearly under one of the dynamic detected regions, and then it is removed from the landmarks map.

8. Conclusions

Traditionally, SLAM algorithms have relied on sparse environment representations. A substantial number of works have appeared during recent years with the objective of reducing the computational complexity of SLAM algorithms. Most of these approaches are based on Kalman filters, which require the map to be modeled in the state-vector form. Consequently, the environment representation maintained by SLAM algorithms is a landmarks map where the landmarks are extracted from sensory information that possesses particular features, and the rest of the information is rejected. One important problem with traditional SLAM algorithms is that the sparse representation maintained cannot provide enough information for autonomous navigation. Despite the large number of works focusing on issues related to the computational complexity of SLAM algorithms, very little work has been done to investigate representations alternative to the classic feature-based.

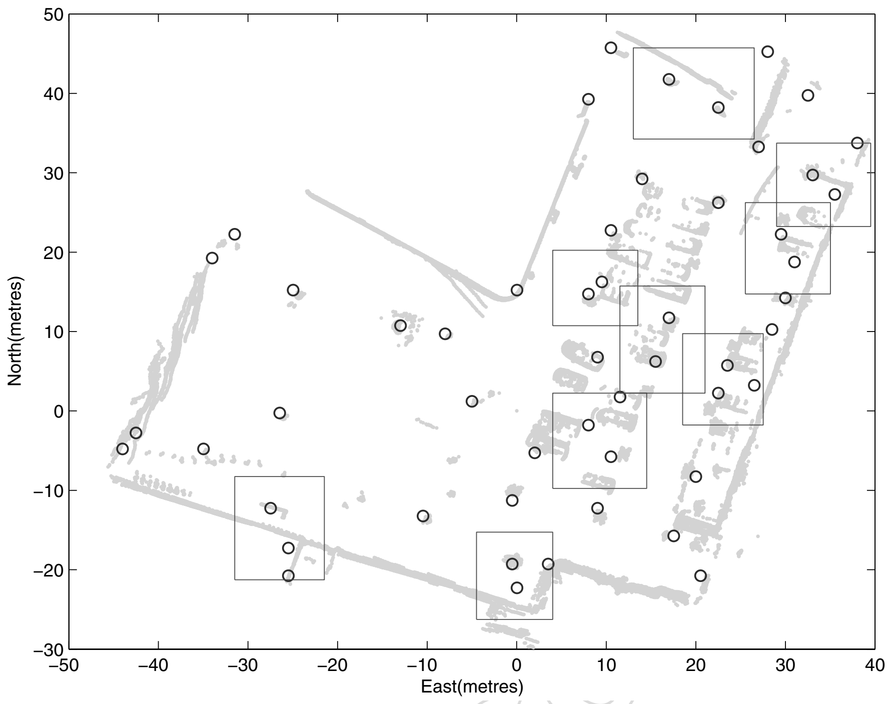
This paper presented the Simultaneous Localization and Dense Mapping (DenseSLAM) problem and an algorithm which is able to obtain dense multi-layer representations. The algorithm presented combines feature maps with other dense metric sensory representations. The approach allows dense multi-layer environment representations to be obtained. Each layer can represent different properties of the environment, such as occupancy, traversability, elevation or each layer can describe the same environment property using different representations. The consistency of the maps was analyzed and it was shown that the algorithm is able to obtain dense maps without losing consistency.

In addition, to be a prerequisite for autonomous navigation, a detailed environment representation also permits improvement of the vehicle localization process; two applications were shown (i) complex landmarks that cannot be observed from only one vantage point were extracted from the dense maps and added into the navigation map, and (ii) the dense representation was used to obtain a map layer of dynamic areas. This map is then used to detect dynamic objects that could potentially be used as valid static landmarks.

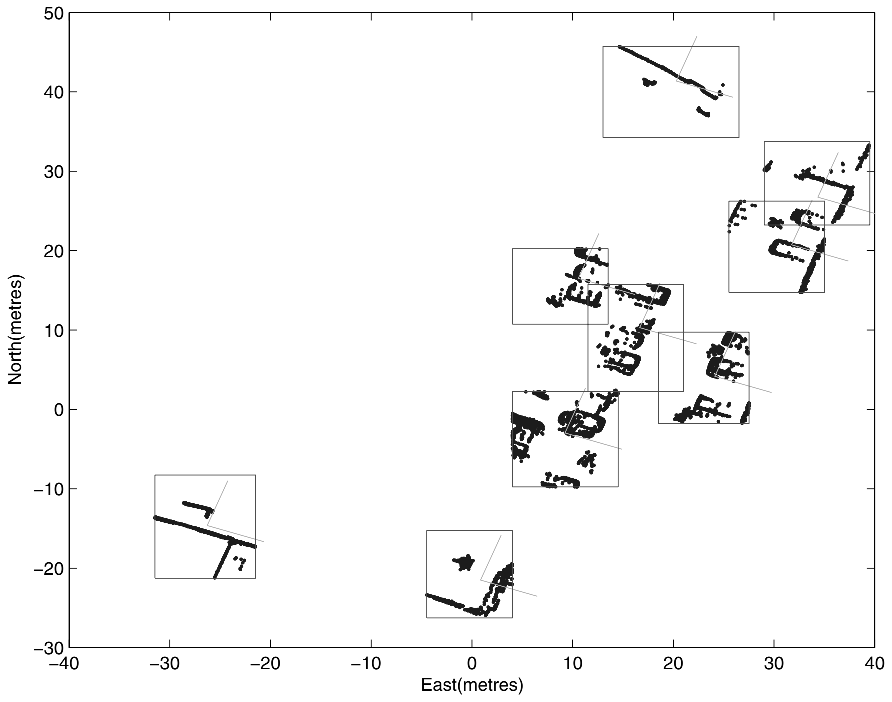
Experimental results in an outdoor environment were presented which validated the algorithm.

Acknowledgments

This work was supported by the ARC Centre of Excellence programme, funded by the Australian Research Council (ARC) and the New South Wales State Government. Also special thanks to QSSL for donating copies of the QNX



(a)



(b)

Fig. 26. Figure (a) shows the HLLs detected. The rectangles represent the bounds used to define the landmarks template. (b) Shows the HLLs map where the points represent the mean of the Gaussians and the axes the position of the local coordinate systems.

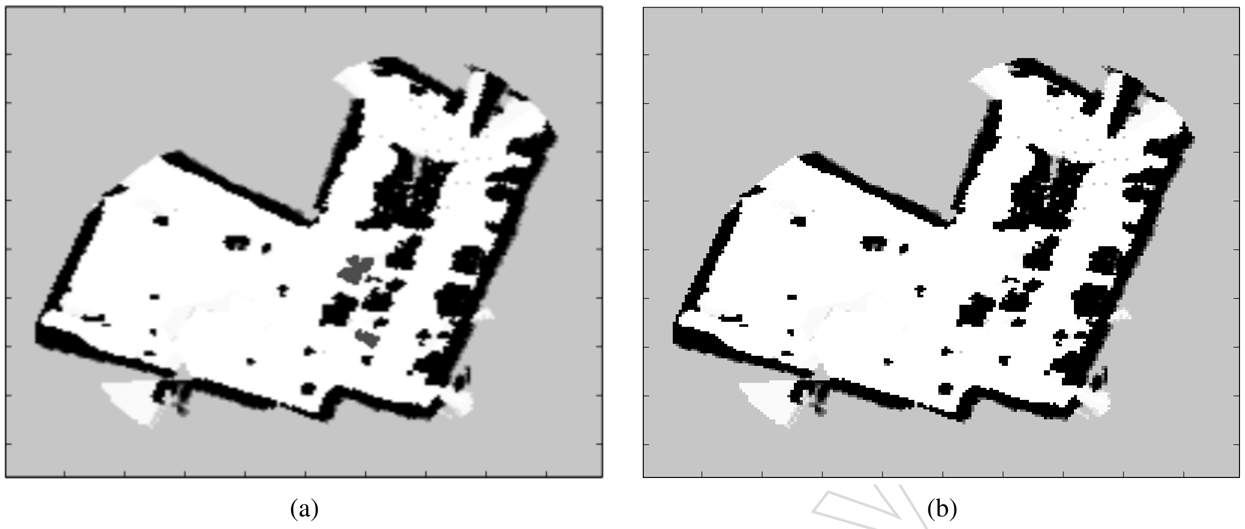


Fig. 27. Slow motion detection algorithm. (a) and (b) show the OG map obtained at two different times. In (b), three cars have been removed from their parking places which are shown in (a) in lighter color. Two spots have been identified as dynamics. The first spot is the detection of two cars that were near and the second spot for the third car removed.

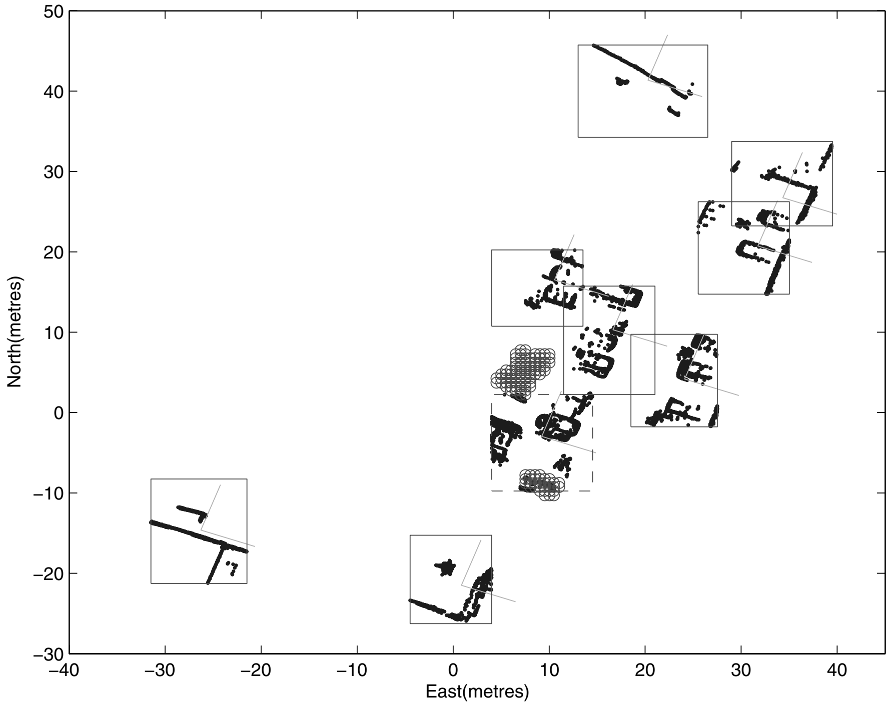


Fig. 28. The lighter points denote the two dynamic areas found. The HLL represented with dashed line is removed from the landmarks map since it is located under one of the dynamic areas detected.

Momentics Professional Development suite used to implement the real time data acquisition system for the vehicle.

References

Bailey, T. 2002. *Mobile Robot Localisation and Mapping in Extensive Outdoor Environments*. PhD thesis, University of Sydney, Australian Centre for Field Robotics.

Biswas, R., Limketkai, B., Sanner, S., and Thrun, S. 2003. Towards object mapping in non-stationary environments with mobile robots. *IEEE International Conference on Intelligent Robots and Systems*, volume 1, pp. 1014–1019.

Bosse, M., Newman, P. M., Leonard, J. J., and Teller, S. 2003. An atlas framework for scalable mapping. *IEEE International Conference on Robotics and Automation*, volume 2, pp. 1899–1906.

Devy, M., Chatila, R., Fillatreau, P., Lacroix, S., and Nashashibi, F. 1995. On autonomous navigation in a natural environment. *Journal of Robotics and Autonomous Systems* (16):5–16.

Elfes, A. 1989a. *Occupancy Grids: A probabilistic framework for robot perception and navigation*. PhD thesis, Department of Electrical Engineering, Carnegie Mellon University.

Elfes, A. 1989b. Using occupancy grids for mobile robot perception and navigation. *IEEE Computer* 22:46–57.

Gibbens, P. W., Dissanayake, G. M. W. M., and Durrant-Whyte, H. F. 2000. A closed form solution to the single degree of freedom simultaneous localisation and map building (slam) problem. *Proceedings of the 39th IEEE Conference on Decision and Control*, volume 1, pp. 191–196.

Guivant, J. 2002. *Efficient Simultaneous Localization and Mapping in Large Environments*. PhD thesis, University of Sydney, Australian Centre for Field Robotics.

Guivant, J. and Nebot, E. 2002a. Acfr, experimental outdoor dataset. http://www.acfr.usyd.edu.au/homepages/academic/enebot/dataset.htm.

Guivant, J. and Nebot, E. 2002b. Improving computational and memory requirements of simultaneous localization and map building algorithms. *IEEE International Conference on Robotics and Automation*, volume 3, pp. 2731–2736.

Guivant, J. and Nebot, E. 2003. Solving computational and memory requirements of feature-based simultaneous localization and mapping algorithms. *IEEE Transactions on Robotics and Automation* (19):749–755.

Guivant, J., Nieto, J., Masson, F., and Nebot, E. 2004. Navigation and mapping in large unstructured environments. *The International Journal of Robotics Research* 23(4):449–472.

Hähnel, D., Triebel, R., Burgard, W., and Thrun, S. 2003. Map building with mobile robots in dynamic environments. *IEEE International Conference on Robotics and Automation*, volume 2, pp. 1557–1563.

Jensfelt, P. and Kristensen, S. 2001. Active global localization for a mobile robot using multiple hypothesis tracking. *IEEE Transactions on Robotics and Automation* 17(5):748–760.

Jose, E. and Adams, M. 2005. Millimetre wave radar spectra simulation and interpretation for outdoor slam. *IEEE International Conference on Intelligent Robots and Systems*.

Knight, J., Davison, A., and Reid, I. 2001. Towards constant time SLAM using postponement. *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 405–413.

Kuipers, B. and Byun, Y. 1991. A robot exploration and mapping strategy based on a semantic hierarchy of spatial representations. *Journal of Robotics and Autonomous Systems* (8):47–63.

Lacroix, S., Mallet, A., Bonnafous, D., Bauzil, G., Fleury, S., Herrb, M., and Chatila, R. 2002. Autonomous rover navigation in a unknown terrains: Functions and integrations. *The International Journal of Robotics Research* 21(10–11):917–942.

Leal, J. 2003. *Stochastic Environment Representation*. PhD thesis, University of Sydney, Australian Centre for Field Robotics.

Leonard, J. J., Rikoski, R. J., Newman, P. M., and Bosse, M. 2002. Mapping partially observable features from multiple uncertain vantage points. *The International Journal of Robotics Research* 21(10):943–975.

McKerrow, P. J. 1993. Echolocation - from range to outline segments. *Intelligent Autonomous Systems*, pp. 238–247.

Montemerlo, M. and Thrun, S. 2004. A multi-resolution pyramid for outdoor robot terrain perception. *Proceedings of the AAAI National Conference on Artificial Intelligence*.

Montemerlo, M., Thrun, S., and Whittaker, W. 2002. Conditional particle filters for simultaneous mobile robot localization and people-tracking. *IEEE International Conference on Robotics and Automation*, volume 1, pp. 695–701.

Moravec, M. P. 1988. Sensor fusion in certainty grids for mobile robots. *AI Magazine* (2):61–77.

Nieto, J., Bailey, T., and Nebot, E. 2005. Scan-slam: Combining ekf-slam and scan correlation. *International Conference on Field and Service Robotics*, pp. 129–140.

Nieto, J., Guivant, J., and Nebot, E. 2004a. Denseslam: The unidirectional information flow (uif). *5th IFAC Symposium on Intelligent Autonomous Vehicles*.

Nieto, J., Guivant, J., and Nebot, E. 2004b. The hybrid metric maps (hymms): A novel map representation for denseslam. *IEEE International Conference on Robotics and Automation*, pp. 391–396.

Pagac, D., Nebot, E., and Durrant-Whyte, H. 1998. An evidential approach to map-building for autonomous vehicles. *IEEE Transactions on Robotics and Automation* 14(4):623–629.

Ribo, M. and Pinz, A. 2001. A comparison of three uncertainty calculi for building sonar-based occupancy grids. *Journal* *of Robotics and Autonomous Systems* 35:201–209.

Smith, R., Self, M., and Cheeseman, P. 1987. A stochastic map for uncertain spatial relationships. *Fourth International Symposium of Robotics Research*, pp. 467–474.

Thrun, S. 2002. Robotic mapping: A survey. *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann.

Thrun, S., Burgard, W., Chakrabarti, D., Emery, R., Liu, Y., and Martin, C. 2001. A real-time algorithm for acquiring multi-planar volumetric models with mobile robots. *The 10th International Symposium of Robotics Research (ISRR'01)*.

Wang, C.-C. 2004. *Simultaneous Localization, Mapping and Moving Object Tracking*. PhD thesis, Robotics Institute, Carnegie Mellon University.

Wang, C.-C., Thorpe, C., and Thrun, S. 2003. Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas. *IEEE International Conference on Robotics and Automation*.

Williams, S. B. 2001. *Efficient Solutions to Autonomous Mapping and Navigation Problems*. PhD thesis, University of Sydney, Australian Centre for Field Robotics.